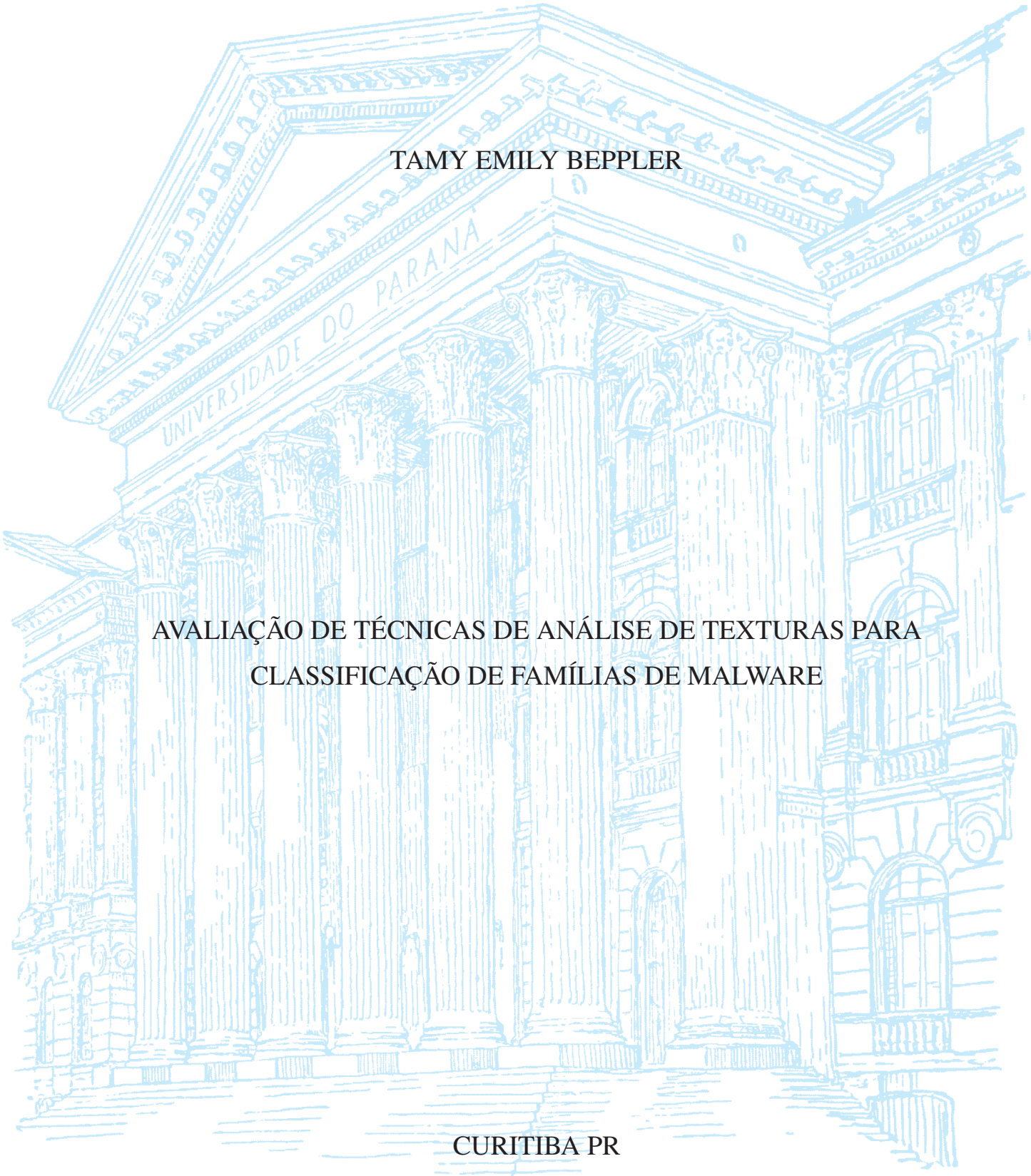
UNIVERSIDADE FEDERAL DO PARANÁ

TAMY EMILY BEPPLER



AVALIAÇÃO DE TÉCNICAS DE ANÁLISE DE TEXTURAS PARA CLASSIFICAÇÃO DE FAMÍLIAS DE MALWARE

CURITIBA PR

2018

TAMY EMILY BEPPLER

AVALIAÇÃO DE TÉCNICAS DE ANÁLISE DE TEXTURAS PARA CLASSIFICAÇÃO DE FAMÍLIAS DE MALWARE

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: André Ricardo Abed Grégio.

CURITIBA PR

2018

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

B481a Beppler, Tamy Emily
Avaliação de técnicas de análise de texturas para classificação de famílias de malware [recurso eletrônico] / Tamy Emily Beppler. – Curitiba, 2018.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2018.

Orientador: André Ricardo Abed Grégio .

1. Computadores – Medidas de segurança. 2. Malware (Software de computador) - Prevenção 3. Redes de computadores - Medidas de segurança. 4. Proteção de dados. I. Universidade Federal do Paraná. II. Grégio, André Ricardo Abed. III. Título.

CDD: 005.8

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **TAMY EMILY BEPPLER** intitulada: **Avaliação de Técnicas de Análise de Texturas para Classificação de Famílias de Malware.**, após terem inquirido a aluna e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 21 de Dezembro de 2018.

ANDRÉ RICARDO ABED GRÉGIO
Presidente da Banca Examinadora (UFPR)

RAFAEL DUARTE COELHO DOS SANTOS
Avaliador Externo (INPE)

FABIANO SILVA
Avaliador Interno (UFPR)

LUIZ EDUARDO SOARES DE OLIVEIRA
Avaliador Interno (UFPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo dom da vida, por me iluminar e proteger em todos os momentos da minha vida.

Aos meus pais, Adalberto e Lizionete, e meu irmão, Adalberto Junior, pelo apoio e incentivo desde o início dessa jornada. Sem esse suporte esse mestrado teria sido apenas um sonho não realizado.

Ao meu professor André Grégio (melhor orientador), pela sua disponibilidade, paciência, por todo auxílio desde o início da orientação e por acreditar mais em mim do que eu mesma. Este trabalho também é mérito seu.

Aos meus amigos, que me motivaram e me acalmaram muitas vezes e por entenderem minha indisponibilidade devido ao tempo dedicado ao projeto. Agradeço em especial a Mariane, o Leonam, o Bruno e o Marcus que acompanharam essa trajetória mais de perto e me ajudaram muito em todas as etapas.

Aos meus familiares e todos aqueles que me encorajaram e contribuíram de alguma forma para a conclusão dessa dissertação.

A CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro .

"A ciência nunca resolve um problema sem criar pelo menos outros dez."
(George Bernard Shaw)

RESUMO

A quantidade de variantes de arquivos maliciosos lançadas diariamente já fez com que a análise manual de *malware* se tornasse inviável há algum tempo. Para isso, foram propostos diversos tipos de análise automatizadas, entre eles a análise estática e a dinâmica, os mais utilizados atualmente. Porém, desenvolvedores de *malware* já conseguiram identificar as falhas de cada um deles e conseguem criar novos arquivos maliciosos que não são nem mesmo detectados pelos antivírus atuais. Para resolver esse problema, pesquisadores têm proposto outros tipos de análise e investido em métodos de classificação mais rápidos e precisos. Neste trabalho de pesquisa, é feito um levantamento bibliográfico sobre o assunto e optou-se por avaliar a classificação utilizando a análise de texturas. Foram selecionadas diversas técnicas para classificação de *malware* usando análise de texturas através de uma revisão sistemática da literatura. Com as técnicas encontradas foram realizados experimentos em um *dataset* da literatura (Malimg) e reaplicados nas amostras de um *dataset* local, mais robusto e semelhante ao cenário do mundo real. Em ambos o algoritmo KNN apresenta os melhores resultados de classificação, mostrando-se a alternativa mais viável em direção à solução do problema de agrupamento de variantes de programas maliciosos em suas famílias corretas através da análise de texturas. As técnicas de classificação usando o descritor global GIST obtêm maior taxa de acerto quando comparadas com o descritor local LBP e o uso de uma escala maior das texturas também apresenta melhor resultado. O *dataset* local atinge resultados bons apenas após uma seleção de dados, apresentando uma discussão sobre o uso de *datasets* não apropriados pela literatura para construção de classificadores genéricos de *malware*. Quanto a resiliência às técnicas de ofuscação utilizadas por criadores de *malware* para descaracterizar um binário, os experimentos ainda apontam como outra falsa teoria sobre a análise de texturas, pois apresenta resultados de classificação bastante ruins mesmo quando utilizadas técnicas bastante simples. A análise de texturas apresenta bons resultados apenas para variantes muito similares, não podendo ser utilizadas num cenário real onde há uma grande variedade de famílias e uso de técnicas bastante sofisticadas de ofuscação.

Palavras-chave: Segurança computacional. Malware. Análise de textura. Classificação de malware. Visualização de malware.

ABSTRACT

The number of malicious software variants released daily turned manual malware analysis into an impractical task a long time ago. Due to that, automated analysis techniques were proposed, such as static and dynamic code analysis, which are the most used nowadays for the malware problem. However, malware authors already identified the shortcomings of each one of these analysis types so as to create new malicious files that are not even detected by current antiviruses. To solve this problem, researchers have proposed other types of analysis and invested in faster and more accurate classification methods. In this research work, I did a bibliographic survey on the subject, which led to the decision of performing classification using texture analysis. Several techniques were filtered to classify malware using texture analysis through a literature systematic review. Experiments were carried out with these techniques applied in a literature dataset (Malimg) and then reapplied to the samples of our lab's malware dataset, more robust and similar to the real world scenario. In both datasets, KNN algorithm presented the best classification results, showing that it is the most viable approach towards solving the problem of grouping malware variants correctly into their families based on texture analysis. The classification techniques using the global descriptor GIST obtain a higher accuracy rate when compared to the local LBP descriptor and the use of a larger scale of the textures also presents better results. The local dataset achieves good results only after a data selection, presenting a discussion on the use of non-appropriate datasets in the literature for building generic malware classifiers. Related to the resilience to obfuscation techniques used by malware writers to deprive a binary, the experiments also point to another false theory about texture analysis, since it presents very bad results even when using fairly simple techniques. The texture analysis presents good results only for very similar variants, and can not be used in a real world scenario where there is a great variety of families and use of quite sophisticated techniques of obfuscation.

Keywords: Computer security. Malware. Texture analysis. Malware classification. Malware visualization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ANN	Artificial Neural Network
AV	Antivírus
CLD	Descritor de Layout de Cor
CNN	Convolutional Neural Network
DT	Decision Trees
DTC's	Decision Tree Classifiers
DWT	Transformada Wavelet Discreta
EUA	Estados Unidos da América
GWT	Transformada Wavelet de Gabor
HTD	Descritor de Textura Homogênea
IoT	Internet das Coisas
KNN	K-Nearest Neighbors
LBP	Local Binary Pattern
MLP	Multilayer Perceptron
NC	Nearest Centroid
PCA	Principal Component Analysis
QT	Quality Threshold
ResNets	Redes Residuais Profundas
RF	Random Forest
SCFG	Gráficos de Fluxo de Controle Estruturado
SGD	Stochastic Gradient Descent
SO	Sistema Operacional
SVM	Support Vector Machine
UFPR	Universidade Federal do Paraná
UPX	Ultimate Packer for eXecutables
VGG16	Visual Geometry Group com 16 camadas

SUMÁRIO

1 INTRODUÇÃO

O desenfreado avanço tecnológico associado ao mundo conectado traz consigo uma série de problemas em relação a segurança computacional. As comodidades atingidas com esse progresso são adquiridas à um preço alto. Arquivos e programas maliciosos têm se espalhado para roubar informações, fazer alterações no sistema e se multiplicar há anos. Apesar da evolução de técnicas para combater esses arquivos, seus criadores também conseguem evoluí-los, esquivando-se das técnicas de detecção. Essa caça de gato e rato está longe de ter um fim e com a diversidade de dispositivos conectados, suas variações estão cada vez mais difíceis de serem identificadas e paradas.

Desde o primeiro *malware* (arquivo ou programa com intenção maliciosa) registrado, os atacantes virtuais se utilizam dos *softwares* maliciosos a fim de prejudicar um sistema. Já no início de 1970 foi detectado um vírus em um computador militar dos Estados Unidos que ganhava acesso, criava cópias de si mesmo e exibia a mensagem *"I'm the creeper: catch me if you can"* (Lab., 2018). Para removê-lo foi criado outro vírus que o encontrava nas máquinas conectadas e o deletava.

Devido aos dispositivos de segurança de rede eficazes, os atacantes virtuais estão focados em atacar diretamente os aplicativos (Conti et al., 2008). Esses atacantes têm criado programas maliciosos cada vez mais sofisticados e que, por muitas vezes, conseguem passar despercebidos pelos atuais antivírus (AV). São criadas variantes de um mesmo *malware* facilmente através de *toolkits* de criação automática e, mesmo com pequenas variações, elas são capazes de enganar as atuais técnicas de detecção (Awad e Sayre, 2016). Com isso, os pesquisadores constantemente buscam novas maneiras de proteger o sistema tentando identificar esses arquivos para entendê-los e pará-los.

Para reconhecer se um arquivo é malicioso ou benígno é necessário identificar quais as suas características. Elas são obtidas através de uma análise do arquivo e, com esse conjunto de características extraído, é possível categorizar os arquivos maliciosos conforme seus objetivos e funções. Essa classificação ajuda a identificar a melhor maneira de defender o sistema das atividades maliciosas do arquivo em questão. Variantes de *malware* possuem apenas alguns atributos diferentes, mas o objetivo malicioso permanece o mesmo, por isso são considerados da mesma família, como será visto no próximo capítulo.

Segundo o relatório de ameaças à segurança na Internet da Symantec (Symantec, 2017), foram observadas mais de 357 milhões de novas variantes de *malware* em 2016 (Figura 1.1). Uma detecção rápida e precisa é essencial para impedir sua proliferação nas máquinas e conseguir evitar uma grande perda.

De 2016 para 2017 pode-se observar a expansão das áreas de ataque. Houve um aumento de 600% no número de ataques através da internet das coisas (IoT); a mineração de criptomoedas (área de maior crescimento em cibercrimes em 2017 (Symantec, 2018)) teve um crescimento de detecções de AV de 8500%; 46% de aumento de novas variantes só de *ransomware* (*malware* que restringe o acesso ao sistema, liberando-o mediante o pagamento do resgate), porém observado surgimento de poucas novas famílias dos mesmos e; 54% variantes de *malware* a mais para dispositivos móveis.

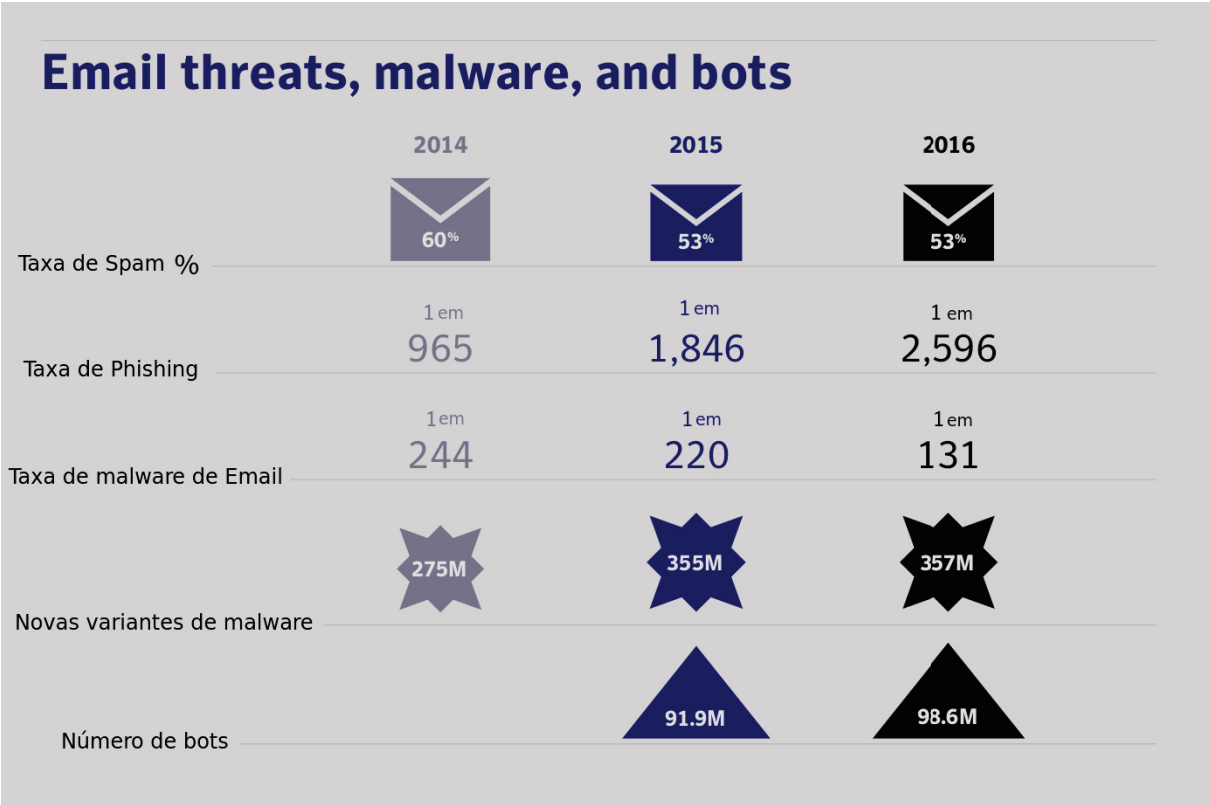


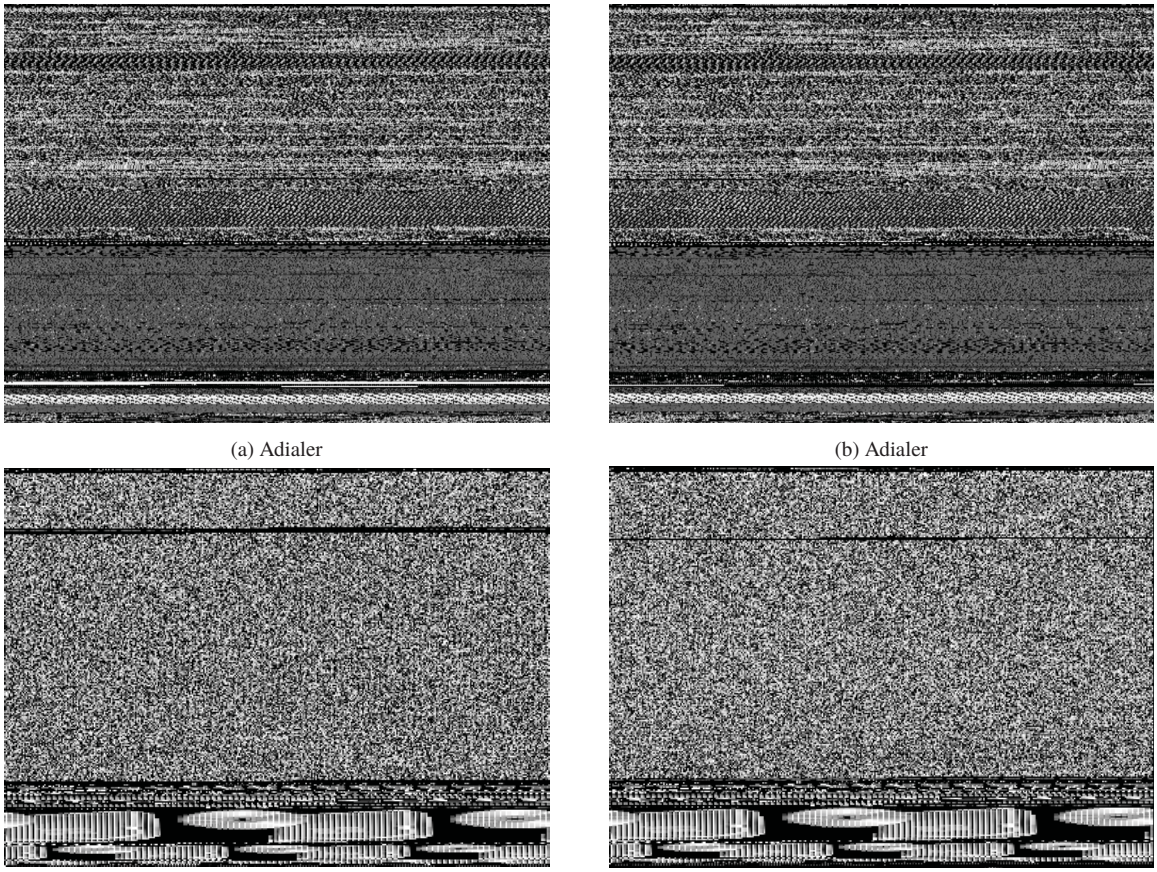
Figura 1.1: Relatório *Symantec* 2016. Adaptado de Symantec (2017).

Conforme Nataraj (2015), os métodos de detecção de *malware* atuais muitas vezes são lentos para reagir aos novos ataques e ameaças. As análises estática e dinâmica já se tornaram sobrecargas computacionais (Nataraj et al., 2013), o que direciona para o uso de uma alternativa diferente de análise, mais rápida e tão bastante precisa. Esse trabalho utiliza a análise de textura, transformando assim o problema de classificação de *malware* em um problema de reconhecimento de imagens.

Técnicas de visualização são cada vez mais usadas para conseguir identificar mais facilmente um problema. Ao converter o binário de um arquivo malicioso em textura, é possível conhecer a aparência desse *malware*. Sabendo qual é o perfil de arquivos maliciosos de determinada família, é possível identificar se uma nova amostra pertence a esta família e, consequentemente, facilita saber como se proteger e tratar o sistema ao receber o determinado arquivo.

Para melhor ilustrar essa aparência da textura de *malware*, a Figura 1.2 mostra o perfil de duas famílias, `Fakerean` e `Adialer`. É possível visualizar como são semelhantes as variantes pertencentes a mesma família (como por exemplo 1.2(a) e 1.2(b)), porém como se diferem entre famílias diferentes (como por exemplo 1.2(a) e 1.2(c)). Ao receber uma terceira amostra de Adialer, não seria muito difícil identificar a qual família esta amostra pertenceria mesmo sem utilizar um computador para comparar cada píxel. Mas quando se fala em um cenário real, com grande quantidade de diferentes famílias existentes, essa tarefa se torna um pouco mais complicada, onde a olho humano se torna mais difícil diferenciar.

Métodos de processamento de imagens são amplamente utilizados em diversas áreas e possuem alta precisão de reconhecimento de padrões (Zhang et al., 2016). Para usufruir dessas vantages, propõe-se o uso de texturas, pois assim é possível realizar a classificação de *malware* com técnicas de reconhecimento de padrões de imagens, buscando por padrões em variantes da



(a) Adialer (b) Adialer

(c) Fakerean (d) Fakerean

Figura 1.2: Textura de variantes de *malware* das famílias Adialer e Fakerean do conjunto de dados Malimg.

mesma família. Para Nataraj (2015), a análise de texturas para classificação de *malware* é um método que funciona para todos os sistemas operacionais (SOs), não sofre com algumas técnicas de ofuscação (que buscam descaracterizar o binário para não ser reconhecido) que sofre a análise estática e é muito mais rápida e com acurácia de classificação comparável a análise dinâmica. O autor realizou experimentos com bons resultados em amostras de diferentes SOs, alguns arquivos foram empacotados, mostrando que a análise de texturas é resiliente a essa técnica de ofuscação e comparou com as análises dinâmica e estática.

Diferentes técnicas de classificação de *malware* baseada em textura foram propostas desde então, muitas delas utilizando o *dataset* disponibilizado por Nataraj et al. (2011a) (Malimg Dataset) com amostras já convertidas em textura. Basicamente é feita uma conversão do binário de *malware* em uma imagem em escala de cinzas e suas características são extraídas por um descritor de texturas. A partir dessas características é possível categorizá-lo na família que pertence com um algoritmo de classificação. Esse trabalho busca avaliar essas técnicas e sua eficiência num cenário mais próximo da realidade, usando um *dataset* local com mais famílias e amostras, além de reavaliar sua resiliência às técnicas de ofuscação.

1.1 DEFINIÇÃO DO PROBLEMA

Para poder interromper uma atividade maliciosa e proteger o sistema, é necessário identificar a intenção maliciosa pelas características do *malware* em questão, extraídas através da análise do arquivo, e em seguida é feita a classificação do mesmo em famílias. Atualmente essa análise e classificação de *malware* possuem um alto custo computacional e, dada a quantidade

de novos arquivos maliciosos lançados diariamente, esse custo precisa ser reduzido, além de ser necessário melhorar a classificação. Visando resolver esse problema, alguns pesquisadores utilizam uma estratégia de análise de texturas de *malware* por ser uma abordagem agnóstica a sistemas operacionais, que não sofre com técnicas de ofuscação comuns, com acurácia similar, porém sendo mais rápida. Essa pesquisa busca responder à pergunta: "em um cenário real, com milhares de exemplares e famílias de *malware*, utilizar unicamente a estratégia de análise de texturas para classificação de *malware* pode resolver esse problema de classificação com as vantagens apontadas na literatura?"

1.2 OBJETIVOS

O objetivo geral desse trabalho é avaliar as atuais técnicas de classificação de arquivos maliciosos baseadas na análise de texturas em um *dataset* mais amplo, semelhante ao cenário real, e validar também o seu uso quanto às técnicas de ofuscação. Os objetivos específicos do trabalho são:

- Realizar uma revisão sistemática para encontrar as técnicas utilizadas na literatura para classificação de *malware* baseadas em análise de texturas.
- Estudar as atuais técnicas de classificação de *malware* por análise de textura já propostas.
- Reimplementar os principais algoritmos utilizados em um *dataset* público usado pela comunidade científica.
- Executar experimentos de classificação de *malware* no *dataset* local ($\approx$ 20 mil amostras) e avaliar a acurácia dos algoritmos em um cenário com maior variedade de famílias e exemplares.
- Verificar o tempo de treinamento e teste para classificação dessas técnicas.
- Avaliar se os algoritmos conseguem classificar famílias de *malware* mesmo quando aplicadas técnicas de ofuscação comuns sob o binário.

1.3 CONTRIBUIÇÃO

As principais contribuições dessa pesquisa são (i) apresentação de uma revisão sistemática da literatura em classificação de *malware* baseada em texturas; (ii) avaliação das atuais técnicas de classificação encontradas na revisão; (iii) identificação de resultados enviesados na literatura que utilizam um *dataset* com seleção de amostras que são mais simples de classificar.

1.4 ORGANIZAÇÃO DO TRABALHO

O restante do trabalho está disposto da seguinte forma: no Capítulo 2 apresenta-se alguns conceitos importantes para compreender o trabalho com base na literatura; o Capítulo 3 mostra o estado-da-arte sobre a análise e classificação de *malware* e aplicações de textura, assim como a revisão sistemática sobre classificação de *malware* baseada na análise de texturas; ao longo do Capítulo 4 é descrita a metodologia do trabalho com os *datasets* utilizados e como ocorre a análise e classificação baseada na análise de texturas; no Capítulo 5 os resultados dos experimentos e uma discussão dos resultados e, por fim, no Capítulo 6, são apresentadas as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados alguns conceitos importantes para a compreensão do restante do trabalho. É explicado sobre o conceito de *malware*, sua análise e classificação, assim como as técnicas utilizadas para fazê-las.

2.1 *MALWARE*

Malware é o nome dado a um arquivo ou programa que possui uma intenção maliciosa: roubar informações, danificar, fazer modificações e/ou se multiplicar, ou seja, causar danos à computadores, redes, servidores, dispositivos móveis, entre outros (Nataraj, 2015). Ele pode ser categorizado em diversas maneiras, de acordo com a plataforma pertencente (Windows, Linux, Android, etc), dividido em classes de acordo com sua função, como por exemplo `Vírus`, `Rootkit` e `Worm`, e em famílias, com uma função ainda mais específica (`Allaple`, `Rbot`, etc.). Essas famílias possuem variantes, que são arquivos maliciosos da mesma família e classe, portanto com a mesma função, porém com alguns atributos diferentes. Reconhecendo a que família pertence uma nova variante é possível identificá-lo e entender seu objetivo malicioso. A Figura 4.1 mostra como um *malware* pode ser classificado.

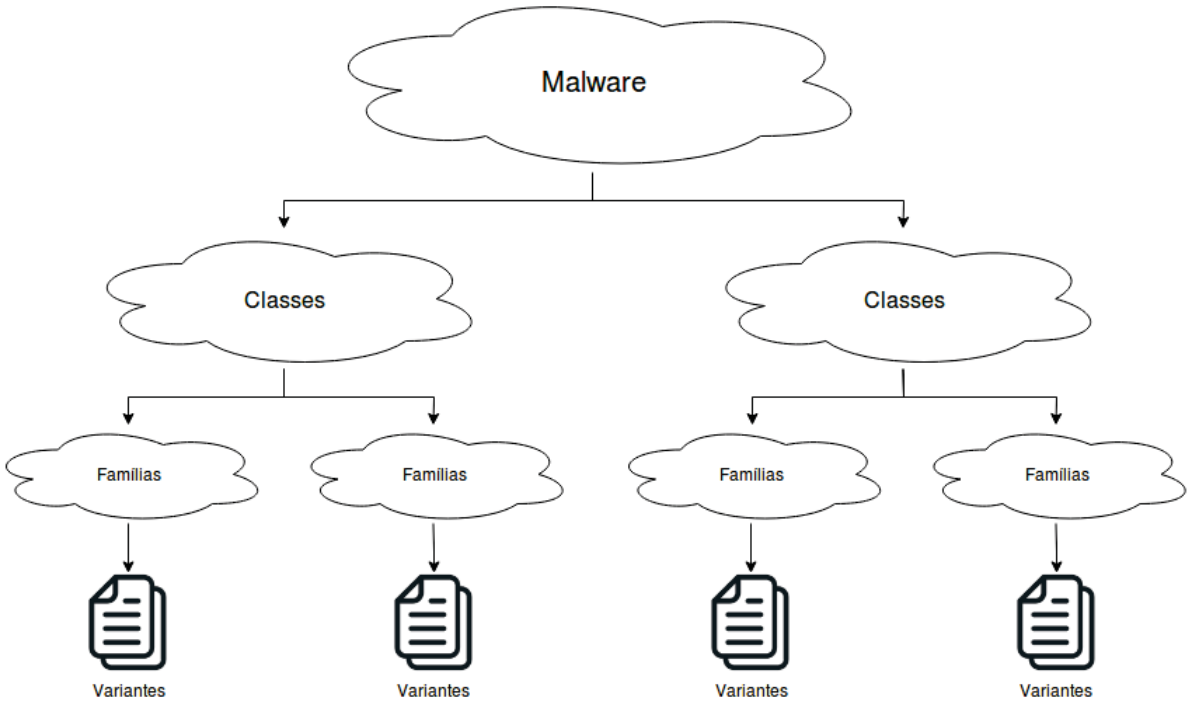


Figura 2.1: Categorização de *Malware*. Adaptado de Nataraj (2015).

Kosmidis (2017) afirma que uma família é um grupo de *malware* onde o código fonte principal do arquivo é o mesmo, as funcionalidades são bastante parecidas e o que é alterado é o

comportamento em cada variante. Como há alterações em seu código, mesmo que pequenas, consequentemente há alterações no binário compilado. Nataraj (2015) explica que quando são utilizadas técnicas de mutação simples, apenas 7 bytes da variante são alterados. Para o autor, mesmo que as variantes continuem executando a mesma função que o *malware* original, como possui essas alterações e atributos diferentes o AV tradicional acaba não as identificando.

Essas abordagens utilizadas atualmente não conseguem identificar as novas variantes de *malware*, mesmo que já se tenha assinaturas de outras variantes conhecidas dessa família (Naidu e Narayanan, 2016). Convertendo o binário em uma imagem em escala de cinzas, segundo Nataraj (2015), as variantes da mesma família de *malware* apresentam semelhança estrutural e visual, o que pode ser uma solução para o problema de classificação de variantes de *malware*, dado que as mudanças são poucas.

De acordo com a classificação desses arquivos em classes, plataforma, famílias e variantes, é atribuído um nome ao *malware*. O nome não possui uma padronização universal, o critério é definido pelo fabricante de AV, podendo ser inconsistente mesmo em versões distintas do mesmo *engine* (AV). Essas diferentes nomeações podem ser um problema na rotulação das amostras usadas em experimentos, produzindo taxas de acerto diferentes de acordo com o rótulo utilizado.

Para tentar amenizar esse problema, nessa pesquisa é utilizada a rotulação proposta por Sebastián et al. (2016), que basicamente faz uma comparação entre os rótulos de diferentes fabricantes de AV, extraídos do VirusTotal[1] (VirusTotal, 2017), e rotula com a categorização mais provável do arquivo.

Tendo um *dataset* rotulado, é possível testar os classificadores, porém, a classificação de *malware* é feita através de uma comparação entre as variantes em busca de padrões semelhantes. Esses padrões e características que são utilizados pra classificação são gerados a partir de uma tarefa de análise do arquivo, tópico abordado na próxima seção.

2.2 ANÁLISE DE *MALWARE*

A análise de *malware*, de acordo com Wagner et al. (2015), é a arte de dissecar o binário, examinar todas as parte do arquivo para entender como ele funciona, como identificá-lo e como se defender ou eliminá-lo. É um processo extremamente importante pois é através dos resultados produzidos nessa etapa que se identifica se um arquivo é malicioso e quais as características de cada tipo de *malware*. Os dois tipos de abordagens mais comumente usados são a análise estática e a análise dinâmica, sendo, no primeiro caso, um exame realizado baseado no binário do arquivo e, no segundo caso, baseado no comportamento apresentado ao executá-lo (Sikorski e Honig, 2012).

A abordagem estática, para Nataraj et al. (2011a), acontece ao desmontar o código e explorá-lo para procurar padrões maliciosos. Para os autores, essa análise possui uma cobertura mais completa que a dinâmica, porém sofre com a ofuscação de código. Conforme Nataraj (2015), acredita-se que quase 80% dos arquivos maliciosos são embalados (o binário é alterado e os dados originais são recuperados na execução). Quando utilizadas essa ou outras técnicas de ofuscação, o executável precisa ser desempacotado e/ou descriptografado (conforme técnica de ofuscação) antes da análise, o que se torna muito mais demorado e aumenta o custo computacional, além de possuir problemas de complexidade intratáveis em alguns casos (como quando usadas técnicas irreversíveis de criptografia).

[1] https://www.virustotal.com/

A análise dinâmica executa o arquivo em um ambiente em geral virtualizado/emulado e apresenta um relatório comportamental com base no rastreamento da execução. Ainda para Nataraj et al. (2011a), ela é considerada mais eficiente que a análise estática e não precisa que o executável seja descompactado ou descriptografado, uma vez que essas atividades são realizadas em tempo de execução. Porém, ela também possui suas desvantagens, pois essa atividade demanda demasiado tempo e consome muitos recursos, o que aumenta os problemas de escalabilidade. E ainda, alguns comportamentos não são observados porque o ambiente não satisfaz as condições necessárias para que eles ocorram (por exemplo, a falta de alguma aplicação específica instalada ou de dados reais de usuários, como fotos). Alguns fabricantes de *malware* também não permitem que o código malicioso seja executado ao detectar um ambiente virtual, simulando uma atividade não maliciosa e evitando assim ser detectado.

Há autores que adotam uma abordagem híbrida, mesclando a análise estática e dinâmica, em uma tentativa de eliminar as desvantagens de ambos os métodos, como mostra Damodaran et al. (2017), mas os autores concluem que os resultados obtidos na classificação permanecem praticamente os mesmos, porém com um alto custo computacional por utilizar as duas abordagens, não considerando vantajosa sua utilização.

Para conseguir se esquivar das técnicas de evasão utilizadas pelos arquivos maliciosos, é interessante abordar um caminho diferente. Segundo Nataraj (2015), é importante o uso de métodos novos e complementares para análise de *malware*, pois tornaria mais difícil e/ou caro para os atacantes escapar de todos os sistemas de detecção. A Tabela 2.1 mostra uma comparação entre esses principais tipos de análise de *malware* e na próxima seção é melhor explicado o que é textura binária e análise de texturas, o tipo de análise usado neste trabalho.

2.3 TEXTURA DE *MALWARE*

Muitas aplicações usam as características extraídas de textura, mas é importante entender o que é textura. Quando se fala de textura de imagem, de acordo com Nataraj et al. (2011b), refere-se a um bloco de pixels com variações de intensidades de padrões repetidos. Para Nataraj et al. (2011a), não existe uma definição comum sobre o significado de textura visual, mas está associada a padrões (repetidos). Na análise de textura de *malware*, um binário de *malware* é convertido em uma representação de imagem para obter os recursos baseados em textura. Exemplos de texturas de *malware* são exibidos na Figura 2.2 extraídas do *dataset* local.

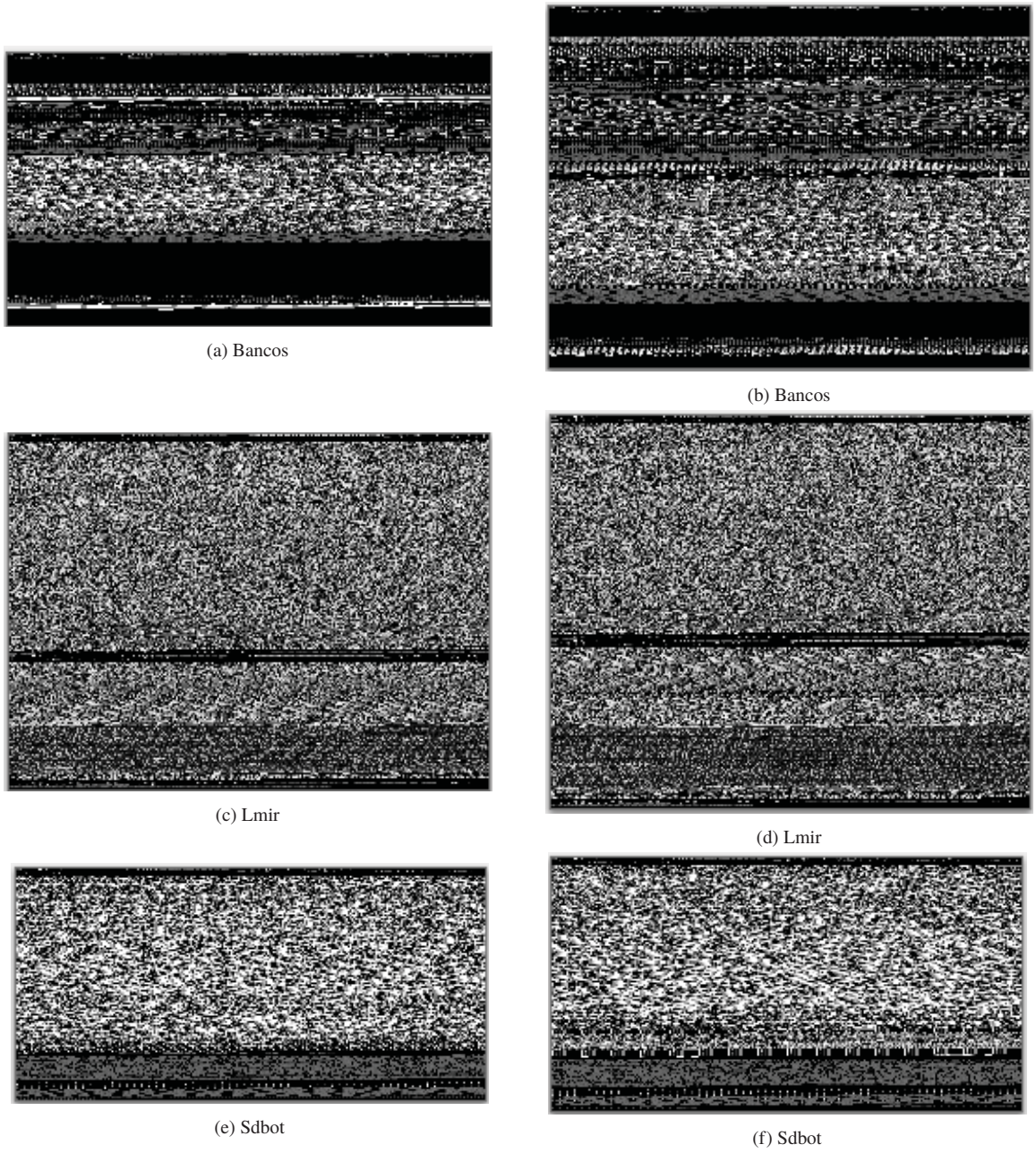
Pode-se observar através da Figura 2.2 que cada família possui algumas características que as diferenciam entre si e que estão presentes na maioria das variantes da mesma família. Sâo as similaridades entre exemplares da mesma família que permitem o agrupamento das texturas e sua classificação.

Nataraj et al. (2011a) explicam que a análise de textura é uma importante área de estudo em visão computacional e ela pode ser explorada pra classificação automatizada. Wagner et al. (2015) afirmam que visualização aumenta a velocidade de detecção de *malware* significativamente. Ao tratar binários de *malware* como imagens, os métodos de reconhecimento de imagem podem ser úteis para melhorar as técnicas de detecção (Zhang et al., 2016). Os autores ainda apontam semelhanças entre o problema de detecção de variantes de *malware* e o problema de reconhecimento de imagens, pois ambos visam reconhecer variações da instância original.

Em Nataraj et al. (2011b), os autores já utilizam técnicas de processamento de imagem para analisar amostras binárias em imagens bidimensionais de escala de cinza e usam os recursos da imagem pra classificação. Isso também é utilizado por Nataraj (2015), Makandar e Patrot (2015b), Makandar e Patrot (2015a), Kosmidis e Kalloniatis (2017), Agarap e Pepito (2017), Kabanga e Kim (2017), Luo e Lo (2017), Makandar e Patrot (2017a) e Yakura et al. (2018a),

Tabela 2.1: Vantagens e desvantagens dos tipos de análise mais comuns.

Análise	Vantagens	Desvantagens
Estática	Não corre risco de infecção, já que não precisa ser executado (Carlin et al., 2017) (Singh, 2017); Observa todos os caminhos de execução possíveis de uma vez (Cepeda et al., 2016) (Willems et al., 2007); Requer menos recursos (Cepeda et al., 2016); Mais precisa do que a abordagem dinâmica quando avaliadas independentemente (Zhang et al., 2016).	Pode falhar quando o binário está ofuscado (Carlin et al., 2017) (Singh, 2017) (Cepeda et al., 2016) (Willems et al., 2007); Precisa desempacotar e descriptografar antes, o que torna caro e as vezes não confiável (Gandotra et al., 2014) (Nataraj et al., 2011b); Consome muito tempo e precisa de um *expert* para analisar (Singh, 2017).
Dinâmica	Mais efetivo e mais útil que estática (Gandotra et al., 2014) (Kosmidis, 2017); Não sofre com técnicas de ofuscação e embalagem. (Singh, 2017) (Cepeda et al., 2016).	Incompleto: só um único caminho de execução é examinado (Carlin et al., 2017) (Singh, 2017) (Cepeda et al., 2016); Comportamento diferente em ambiente virtual, emulado, simulado ou virtualizado (Gandotra et al., 2014) (Carlin et al., 2017) (Singh, 2017) (Nataraj, 2015) (Kosmidis, 2017); Sobrecarga de tempo de execução (ferramentas estáticas podem levar centenas de vezes menos tempo) (Carlin et al., 2017); Tempo intenso e consumo de recursos, elevando problemas de escalabilidade (Gandotra et al., 2014) (Cepeda et al., 2016); Um pequeno problema pode infectar o computador real e toda a rede (Singh, 2017); Menos robusta quando um grande dataset (Nataraj et al., 2011b); Analisa apenas uma única execução de *malware* por vez (Willems et al., 2007).
Híbrida	Supera as limitações da análise estática e dinâmica (não sofre com técnicas de evasão e de ofuscação)(Mathur e Hiranwal, 2013); Combina as vantagens da análise dinâmica e estática (Uppal et al., 2014).	Complexidade das técnicas torna difícil discernir o benefício real (Damodaran et al., 2017); Improvável que seja superior a dinâmica; (Damodaran et al., 2017); Não oferece melhoria consistente em relação a estática (Damodaran et al., 2017).
Textura	Não exige desmontagem ou execução, pois resiste às técnicas de ofuscação (Kosmidis, 2017) (Nataraj et al., 2011b); Pode produzir resultados parecidos à análise dinâmica, mas a 1/4000 quantidade de tempo (Nataraj et al., 2011b) (Nataraj, 2015); Precisão quase igual a estática, porém cerca de 100 vezes mais rápida e precisa cerca de 170 vezes menos recursos (Nataraj, 2015); Agnóstica a SO (Nataraj, 2015).	Pode sofrer com inserção de código lixo entre seções ou troca de ordem das seções (Nataraj et al., 2013); Não identifica variantes funcionalmente semelhantes que possuem estruturas diferentes (Nataraj et al., 2013).



(a) Bancos

(b) Bancos

(c) Lmir

(d) Lmir

(e) Sdbot

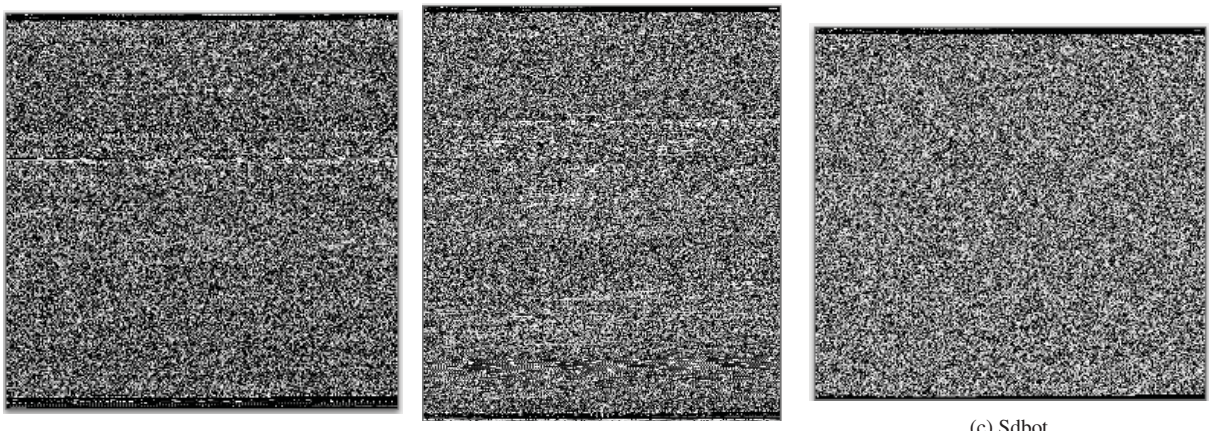
(f) Sdbot

Figura 2.2: Textura de variantes de *malware* de diferentes famílias do conjunto de dados local.

observando que as variantes da mesma família de *malware* apresentam semelhança estrutural e visual e através das técnicas de similaridade de imagem é possível capturar essa semelhança entre elas.

O uso da técnica baseada em textura na classificação de *malware* resiste às técnicas de embalagem, além disso, categoriza com êxito uma grande quantidade de *malware* com fragmentos criptografados e não criptografados, porém os classifica como família diferente (Kosmidis, 2017). Conforme Nataraj et al. (2011b), essa nova abordagem de análise de *malware* não requer desmontagem, descompilação ou execução do binário e, segundo o autor, permite capturar padrões geralmente não capturados. Além disso, funciona independente de SO para o qual o *malware* foi projetado (Nataraj, 2015), o que é uma grande vantagem quanto aos outros métodos de análise.

Apesar dos autores alimentarem grandes expectativas para a abordagem, nem todas as variantes de *malware* de uma determinada família são tão similares quanto as demais. Criadores de *malware* tentam se esquivar da detecção há anos e desenvolveram amostras com poucas características semelhantes. Na Figura 2.3 são apresentadas outras variantes das mesmas famílias já exibidas na Figura 2.2. Na figura anterior elas eram facilmente diferenciáveis, porém outras amostras das mesmas famílias mostram que nem todas elas possuem essas características tão marcantes, o que dificulta o processo de classificação.



(a) Bancos (b) Lmir (c) Sdbot

Figura 2.3: Textura de novas variantes de *malware* de diferentes famílias do conjunto de dados local.

Para realizar a análise de textura, é extraído um vetor de características por um descritor da imagem, que filtram as características mais importantes da textura. A escolha do descritor pode interferir diretamente no resultado final da classificação. Os descritores mais frequentemente utilizados na literatura podem ser divididos basicamente em descritores locais e descritores globais. Os globais analisam a imagem como um todo, portanto atribuem características da textura seguindo um padrão exato. Já os descritores locais dividem a imagem em blocos, e registram características desses blocos da imagem.

Nas pesquisas de Nataraj et al. (2011b) e Kosmidis e Kalloniatis (2017), o descritor global de imagens GIST (Oliva e Torralba, 2001) foi utilizado para essa etapa de extração do vetor de características. Em Nataraj (2015), foi feita uma breve comparação entre os descritores de Textura Homogênea (HTD) (Manjunath et al., 2001), de Layout de Cor (CLD) (Manjunath et al., 2001) e o GIST e o autor optou por usar o GIST por possuir uma descrição mais rica e oferecer melhor precisão. De acordo com Oliva e Torralba (2001), esse descritor utiliza um conjunto de dimensões perceptivas (naturalidade, abertura, rugosidade, expansão, robustez) estimadas usando informações espectrais e grosseiramente localizadas. Ainda para os autores, um espaço multidimensional é gerado e as cenas que possuem as mesmas categorias semânticas são projetadas juntas. Esse modelo desconsidera informações específicas, pois uma análise global seria suficiente para identificar as categorias semânticas. De acordo com Nataraj (2015), são passados na imagem diferentes filtros de várias escalas e orientações, dividido em blocos e a média desses blocos formam o descritor.

Como visto em Nataraj et al. (2011a), utilizar um descritor global pode ser uma desvantagem, pois uma mudança de seções em um binário ou ao adicionar grande quantidade de dados redundantes faria com que o exemplar não fosse classificado corretamente. Os próprios autores sugerem a possibilidade do uso de um descritor local para corrigir essa falha. Com isso, para essa pesquisa, optou-se também por utilizar um descritor local de imagens para comparação.

O LBP (*Local Binary Pattern*) (Ojala et al., 2002) é provavelmente o mais conhecido descritor de imagens local e bastante utilizado por ser simples e eficiente (Simon e Uma, 2018). Ele foi utilizado em Luo e Lo (2017) e comparado com o GIST na classificação de *malware* baseada em texturas apresentando uma taxa de acerto mais elevada que o descritor global. A Figura 2.4 exibe o funcionamento desse descritor. Analisado em grade 3x3, é usado como limiar o píxel central. Os seus oitos vizinhos são então limiarizados (quando seu valor for menor que o central, é atribuído valor 0, caso contrário, 1), gerando uma nova matriz. Em seguida essa nova matriz é multiplicada pela matriz de pesos e seus resultados são somados. O vetor de característica é extraído calculando o histograma da imagem (Luo e Lo, 2017).

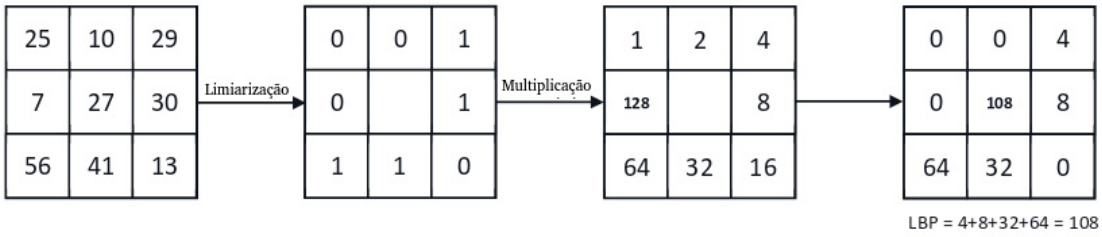


Figura 2.4: Descritor LBP. Adaptado de Luo e Lo (2017).

Após a extração de características através da análise de texturas dos exemplares, é possível realizar a classificação de *malware* em famílias, isto é, categorizar as amostras identificadas como arquivos maliciosos em suas respectivas famílias. As amostras são identificadas como arquivos maliciosos através da detecção de *malware*, onde o binário é rotulado como executável benígno ou malígno; já a classificação de *malware* é o processo de descobrir a que classe ele pertence (Singh, 2017). Este trabalho foca na classificação, portanto na próxima seção há uma explicação apenas sobre a classificação de *malware*.

2.4 CLASSIFICAÇÃO DE *MALWARE*

Classificação é uma técnica onde um objeto deve ser identificado e categorizado em uma classe pré-definida (Kosmidis, 2017). Nesse caso, o objeto em questão é o arquivo malicioso e a classe é a família a qual ele pertence. O objetivo dessa pesquisa é identificar se o *malware* que está sendo analisado se trata de uma variante das famílias já conhecidas. Identificar a família de um *malware* desconhecido é crucial para poder entendê-lo e pará-lo (Nataraj, 2015).

Kosmidis (2017) afirma existir uma necessidade de maiores estudos em relação a classificação de *malware*, pois cada conjunto de dados é um problema diferente a ser tratado. Experimentos que atingem alta taxa de acerto em um determinado *dataset*, podem obter resultados bastante ruins em outro. Isso pode ter relação com o método de rotulação de cada *dataset*, com os tipos de arquivos, com diferentes técnicas de ofuscação entre outros fatores.

A classificação binária de *malware*, conforme Nataraj et al. (2011b), objetiva descobrir se uma nova amostra pertence a uma família de binários conhecida ou se ela se trata de uma nova descoberta. Os autores asseguram que a técnica de classificação supervisionada baseada na análise de textura pode ser altamente escalável e é comparável à análise dinâmica em termos de precisão. Na literatura já foram usados diversos classificadores e neste trabalho optou-se pela utilização dos mesmos para comparação. Nesta seção é dada uma breve explicação sobre eles. Um dos mais simples é o **K-*Nearest Neighbors* (KNN)** (Fix e Hodges Jr, 1951), que é o método

mais básico baseado em instância, pois assume que todas as instâncias correspondem a pontos no espaço e relaciona-as com suas vizinhas (Mitchell, 1997). Para os experimentos, esse tempo de construção dos pontos no espaço é considerado como o tempo de treinamento. A distância pode ser medida de diferentes maneiras, porém a distância euclidiana é a mais comum. Para definir o valor de k-vizinhos é necessário prudência, pois um número menor fica sensível a ruído, porém um número maior pode englobar elementos de outras classes, pois a inclusão de ruídos, faz os critérios ficarem menos distintos. Preferencialmente se escolhem valores ímpares para evitar empates na votação.

Outro algoritmo bastante conhecido para classificação é o ***Decision Trees* (DT)** (Quinlan, 1986), ou Árvores de Decisão. Para Safavian e Landgrebe (1991), os *Decision trees classifiers* (DTC's) são capazes de quebrar um problema complexo de decisão em um conjunto de sub-problemas mais simples, a estratégia "dividir para conquistar". Dividindo-os, cria-se então uma árvore onde em cada nó são armazenadas regras e nas folhas, uma decisão. Para Kosmidis (2017), cada filho da árvore é considerado um rótulo de classe, mas também muitas folhas podem ter o mesmo rótulo. Baseado nas árvores de decisão surgiu o ***Random Forest* (RF)** (Ho, 1995). Segundo Breiman (2001), as florestas aleatórias são uma combinação de árvores de decisão, onde cada árvore depende dos valores de um vetor aleatório. Trata-se de uma ferramenta efetiva na previsão de classes e inserindo o tipo certo de aleatoriedade faz com que sejam classificadores muito precisos. A implementação do `scikit-learn`[2] (Pedregosa et al., 2011) combina os classificadores fazendo uma média da probabilidade de predição, ao invés de deixar o classificador optar por uma única classe (Pedregosa et al., 2011).

Outro classificador simples e rápido, conforme Levner (2005) é o ***Nearest Centroid* (NC)** (McIntyre e Blashfield, 1980). Assume-se que as classes alvo correspondem a clusters únicos e usa a média (centróide) deles para determinar a classe de um novo ponto de amostra. Para Kosmidis (2017), cada família de *malware* é representada pelo centróide de suas amostras e as amostras de teste são classificadas na família cujo centróide seja mais próximo. Em (Makandar e Patrot, 2015b) os autores utilizam ***Support Vector Machine* (SVM)** (Cortes e Vapnik, 1995), que recebe vetores de entrada que são mapeados em um espaço de características de grande dimensão e uma superfície de decisão linear é construída (Cortes e Vapnik, 1995). Basicamente cria um hiperplano (como uma linha de separação) entre padrões de classes diferentes (Makandar e Patrot, 2015b).

O ***Stochastic Gradient Descent* (SGD)** (Robbins e Monro, 1985) também é um algoritmo de classificação linear eficiente e simples, que realiza classificação em grande escala com bons resultados (Kosmidis, 2017). É uma boa estratégia para espaços de busca muito grandes ou infinitos (Mitchell, 1997). De acordo com Avila (2017), ele busca minimizar a função de perda (aquela que mede o erro empírico da classificação), e ao invés de avaliar todas as amostras por iteração, avalia uma única aleatória e atualiza os parâmetros. Conforme Mitchell (1997), atualizar os pesos incrementalmente e fazer o cálculo de erro para cada modelo individual, tenta reduzir o problema de ficar preso em mínimos locais. Outro classificador linear utilizado é o ***Perceptron*** (Rosenblatt, 1957). De acordo com Kosmidis (2017), o *Perceptron* é usado basicamente para decidir se um vetor de dados de entrada pertence a uma classe ou outra e tenta classificar usando uma função de predição linear combinada com um conjunto de pesos. Para o autor, se o conjunto de treinamento não for linear, ele nunca encontrará uma solução ideal, mas se for, deve encontrar uma solução aceitável.

As redes neurais artificiais, ou ***Artificial Neural Network* (ANN)** Vemuri (1988), estão cada vez mais sendo utilizadas para classificação. Simula um cérebro humano (neurônios interligados por onde a informação é transferida e conforme se tem novas experiências, novas

[2]https://scikit-learn.org/

conexões se formam) organizando os neurônios em camadas, onde se inicia na camada de entrada e termina na de saída, podendo existir diversas camadas entre elas (camadas ocultas). A cada conexão entre camadas, a informação é modificada (dado um peso e função) antes de ser transmitida ao próximo neurônio (Singh, 2017). O autor ainda afirmam que os neurônios artificiais também são chamados de *Perceptron* e, quando há múltiplas camadas de neurônios, a rede é chamada ***Multilayer Perceptron* (MLP)** (Rumelhart et al., 1985).

O classificador MLP, ou perceptron multicamadas, é um modelo que representa um mapeamento não-linear entre um vetor de entrada e um de saída (Gardner e Dorling, 1998). Os autores afirmam que a arquitetura de um MLP é variável, mas, em geral, consiste em várias camadas. Kosmidis (2017) considera três: de entrada, as camadas ocultas e de resultados. De um nível a outro, ou uma camada a outra, todos os nós são conectados com todos os nós da camada anterior e posterior. Os MLPs têm a capacidade de aprender através do treinamento e aprendem de forma supervisionada. Ter uma grande rede de camadas ocultas dá flexibilidade ao sistema, mas, para Kosmidis (2017), uma desvantagem é a ineficiência para lidar com dados ilimitados.

Para aprender tarefas mais difíceis, é necessário inserir um número maior de camadas ocultas e o número de camadas representa a profundidade da rede neural. Chamamos de *Deep Neural Network* um modelo "profundo", com muitas camadas. Costumam funcionar bem em imagens menores, pois aumentam demais o custo computacional em imagens maiores (Singh, 2017). Segundo o autor, para cobrir as desvantagens atribuidas às *Deep Neural Networks*, surgiu a ***Convolutional Neural Network* (CNN)** (LeCun et al., 1989), que utiliza camadas convolucionais (são aplicados diversos filtros para extrair diferentes características da imagem), de *pooling* (reduz dimensionalidade, mantendo apenas características mais significativas) e termina com uma camada totalmente conectada.

De acordo com Singh (2017), conforme a quantidade de famílias aumenta, essa tarefa fica muito difícil e devido a isso, a Microsoft apresentou as **ResNets** (redes residuais profundas) (He et al., 2016) com 152 camadas, que reduz o problema de degradação inserindo as entradas de uma camada nas entradas das camadas posteriores. He et al. (2016) também apresentam ResNet com 50 camadas que atinge bons resultados porém com uma quantidade consideravelmente menor de camadas, que foi utilizado neste trabalho e também por Rezende et al. (2017).

2.5 CONCLUSÃO

Malware é um software malicioso e pode ser classificado em diversas classes e famílias, onde cada *malware* similar e que pertencente a uma mesma família é considerado uma variante dela. É muito importante saber a que família ele pertence para saber como combatê-lo.

Antes de realizar a classificação é necessário primeiramente analisar os arquivos. A análise de *malware* normalmente é do tipo estática ou dinâmica. Ambos possuem prós e contras e, por serem tão comuns, os atacantes conseguem driblá-las para que os arquivos não sejam detectados. Por isso opta-se por uma abordagem diferente, não conhecida pelos criadores de *malware* e que não sofra com as técnicas de ofuscação e evasão. Reconhecendo os benefícios do uso de reconhecimento de imagens e o uso de textura binária para classificação em diversas outras áreas, acredita-se que o uso da textura de *malware* para classificação seja um grande aliado.

A textura binária é composta de pixels de diferentes intensidades, porém com padrões das variações de intensidade repetidos. Técnicas de processamento de imagens capturam essas similaridades. Conforme Kosmidis (2017), o uso da análise de textura na classificação de *malware* resiste às técnicas de embalagem, funciona para diferentes SOs e possui uma precisão comparável à análise dinâmica. Para extrair as características da textura é usado um descritor

de texturas. Foi apresentado um descritor global e um local que deve cobrir as desvantagens encontradas na pesquisa de Nataraj (2015).

Com isso, a classificação de *malware* pode categorizar as amostras em suas famílias. Existem muitos classificadores já propostos na literatura e foram selecionados alguns já utilizados para classificação de texturas de *malware* que foram brevemente explicados e utilizados nos experimentos desse trabalho.

Com esses conceitos em mente, é possível seguir ao próximo capítulo onde se retrata o atual estado de arte sobre a classificação de *malware*, incluindo análises estática, dinâmica e de textura.

3 TRABALHOS RELACIONADOS

Para entender o atual estado da arte e poder avaliar os estudos já publicados de classificação de *malware* baseada em textura, é relevante conhecer as diferentes técnicas empregadas para análise e classificação de *malware*, bem como a aplicação de texturas para classificação em diferentes cenários, que são exibidas neste capítulo. Para seleção das técnicas utilizadas na literatura a serem avaliadas, também foi realizada uma revisão da literatura sobre a classificação de *malware* baseada na análise de texturas, que será exibida a seguir.

3.1 ANÁLISE DE MALWARE

Conforme visto no capítulo anterior, os dois tipos de análise mais utilizados são a dinâmica e a estática. Nesta seção serão exibidas algumas pesquisas recentes que utilizam principalmente estas análises.

Em Willems et al. (2007), os autores optam por utilizar a análise dinâmica e implementam o CWSandBox, uma ferramenta de análise de malware para sistemas Win32. Para analisar, rodam as amostras num ambiente simulado e verificam as chamadas do sistema para gerar automaticamente um relatório detalhado. Segundo os próprios autores, usam o CWSandBox para rastrear e monitorar todas as chamadas de sistema relevantes. O CWSandbox só descreve o comportamento visível do malware e seu uso pode causar danos a outras máquinas que estejam conectadas à rede.

Em Camilo et al. (2016) os autores utilizam visualização e análise dinâmica, explorando as relações do tráfego da rede produzido por amostras de malware. Baseiam-se na correlação de comunicações de *malware* por grafos, buscando isomorfismos entre eles e conseguiram bons resultados, concluindo que a busca por isomorfismo é uma ótima ferramenta para agrupar campanhas de *malware* (amostras conectadas ao mesmo servidor e domínios diferentes recebendo as mesmas solicitações). A limitação encontrada é a evasão do *malware* em ambientes monitorados de onde é obtido o tráfego da rede através da análise dinâmica.

Grégio et al. (2011) apresentam BehEMOT, um sistema de análise dinâmica automática de *malware* que identifica comportamentos maliciosos observando as ações realizadas, monitorando através das chamadas de sistema mais relevantes e tráfego de rede. O sistema apresenta um relatório mais compacto e mais fácil de entender que o Anubis e CWSandBox, por exemplo.

A análise dinâmica é utilizada por diversos autores e é considerado um método muito eficiente para caracterizar *malware*. Diferentes técnicas e ferramentas são usadas para essa tarefa, estas podem ser vistas em Egele et al. (2008) que apresentam um *survey* dessas técnicas usadas para análise dinâmica.

Uma das desvantagens da análise dinâmica em geral, conforme Willems et al. (2007), é que ela analisa apenas uma única execução de malware por vez. E como visto anteriormente, os atacantes já estão preparados para combater a análise dinâmica através de técnicas onde detectam o ambiente simulado e não executam o programa malicioso ali ou imitam um comportamento diferente (Awad e Sayre, 2016) (Nataraj et al., 2011a) (Rieck et al., 2008) (Zhang et al., 2016).

De acordo com Zhang et al. (2016), ao avaliar independentemente, a abordagem estática é mais precisa do que a abordagem dinâmica. A análise estática permite prever o comportamento do programa em todas as condições de execução (Awad e Sayre, 2016). Naidu e Narayanan (2016) realizam uma investigação de abordagens para a geração automática de assinaturas visando a identificação de novas variantes polimórficas. Verificam os efeitos de usar diferentes matrizes de substituição em um método baseado em *strings* para a geração automática de assinaturas. Usam o algoritmo de Smith-Waterman de correspondência de cordas que pode identificar as variantes polimórficas conhecidas de um determinado *malware*. Acreditavam que apenas a análise semântica revelaria semelhanças entre variantes da mesma família para a geração efetiva de assinaturas. Para algumas variantes o algoritmo tem uma boa acurácia, enquanto para outras, não pode identificar. Acabam sendo inconclusivos quanto a eficácia da abordagem.

Uma comparação de diferentes técnicas de aprendizado de máquina para análise de *malware* através da análise estática é exibida em Nath e Mehtre (2014). A análise estática possui a desvantagem de ser sensível a técnicas de ofuscação como embalagem, criptografia, compressão, inserção de código lixo e permuta de código (Anderson et al., 2012) (Awad e Sayre, 2016) (Cepeda et al., 2016) (Nataraj, 2015) (Nataraj et al., 2011a) (Nataraj et al., 2011b) (VanHoudnos et al., 2017) (Willems et al., 2007)).

Um pequeno *survey* de diferentes ferramentas e técnicas de análise de *malware* é apresentado por Uppal et al. (2014) e técnicas de análise de *malware* são vistas em Gadhiya e Bhavsar (2013).

Uma detecção de anomalia estatística para *malware* incorporado é proposta em Shafiq et al. (2008). Os autores propuseram um detector Markov n-gram que fornecia melhores taxas de detecção e falso positivo do que o esquema de detecção de *malware* incorporado existente até então. A taxa de entropia de Markov n-grams no arquivo fica significativamente perturbada onde há incorporação de código malicioso. Ele é capaz de localizar a posição da infecção no arquivo infectado. Concluem que pode atingir alta precisão quando utilizado com outro AV, porém perde desempenho quando há dados criptografados. Possui as desvantagens de ter alta taxa de falso positivo para alguns tipos de arquivos, sobrecarga computacional e pode ser enganado por ataque de mímica (simulando estatística benígna).

É feita uma comparação entre as análises estática, dinâmica e híbrida em Damodaran et al. (2017) com um propósito maior de identificar se há uma vantagem inerente no uso da análise híbrida. Utilizam técnicas com base em *API call sequence* e *opcode sequence* e usam *Hidden Markov Models*. Por fim, concluem que a análise dinâmica baseada em chamadas de API é mais eficaz e a estática baseada em *opcode* (código de operação) é quase tão eficaz na maioria dos casos, mas os autores consideram bastante improvável que a abordagem híbrida seja superior à detecção totalmente dinâmica e quando comparada com a estática, também não oferece uma melhoria consistente.

Em Han et al. (2013), os autores analisam visualmente o *malware* transformando o binário em matrizes de imagens com resultados que mostram que podem efetivamente classificar famílias de *malware* e um tempo médio para calcular as semelhanças de cerca de 2.4 ms. Obteve-se semelhança média de 0.984, quando dentro da mesma família e entre famílias diferentes, 0.309. O método gera pixels coloridos RGB em matrizes de imagens conforme a informação binária extraída pela análise estática. Calculou-se as similaridades pela correspondência de área seletiva. Em Wagner et al. (2015) é realizado um levantamento de pesquisas que utilizam técnicas de visualização para análise de *malware*. Os autores afirmam que visualização aumenta a velocidade de detecção de *malware* significativamente.

Na próxima seção é exibido o atual estado da arte da classificação de *malware* que utiliza esses diferentes tipos de análise.

3.2 CLASSIFICAÇÃO DE MALWARE

Diversas técnicas de análise são usadas para gerar insumos que permitam realizar a classificação de *malware*. Em Awad e Sayre (2016) os autores optam pela análise estática visando criar um método preciso e automático que acelere a classificação de variantes em famílias. Nessa pesquisa, os autores agrupam variantes de malware com base nos gráficos de fluxo de controle estruturado (SCFG) das instâncias, gerados pela ferramenta de engenharia reversa de análise estática binária Hyperion. Para descobrir a similaridade, calculam a distância de edição de *string* entre as representações do SCFG e então, usam o algoritmo de clusterização *Quality Threshold* (QT) para agrupar seqüências semelhantes de acordo com um limite adequado. Atingem uma média de 94% de precisão, porém usando apenas um conjunto pequeno de amostras (303 instâncias de malware) descompactadas, o que pode interferir no resultado.

Alguns autores afirmam que para a análise estática, há dois modos de representar *malware*, visão hexadecimal e assembly. Ahmadi et al. (2016) extraem características usando ambas e, para utilizar as características mais relevantes, fazem uso de um algoritmo baseado na técnica *Forward Stepwise Selection*. Para classificador, os autores analisaram a avaliação feita por pesquisadores que mostra que, analisando aproximadamente 180 classificadores, usando diferentes famílias e bases de dados, *Random Forest* e SVM são os classificadores que apresentam melhores resultados, porém, como nas últimas competições Kaggle os vencedores usavam em sua maioria a técnica XGBoost (que na maioria dos casos antingia melhores resultados que *Random Forest*), esse foi o escolhido e alcançou uma acurácia de aproximadamente 99,8% na base do *Microsoft Malware Challenge*.

Em Rieck et al. (2008), os autores fazem uso do CWSandBox pra análise dinâmica onde o comportamento do *malware* é monitorado em um sandbox e em seguida, através de uma base de *malwares* rotulados por um AV, um classificador de comportamento é treinado usando SVM e, por fim, classifica as características dos modelos de comportamento, alcançando quase 70% de acurácia. Os autores acreditam que o motivo desse resultado seja pela execução única do arquivo e problemas de rotulação do AV. Já Rieck et al. (2011), realizam agrupamento e classificação de malware, obtendo melhores resultados. Sua classificação usando *nearest prototype classification (Navy)* atinge um *F-measure* 93 a 95% para *malwares* conhecidos e 96 a 99% para não conhecidos no *dataset* Malheur.

Shaid e Maarof (2014) usam visualização na análise dinâmica, coletando as chamadas de funções de API e as mapeando em cores RGB. Para isso foi definido que as cores quentes representariam os comportamentos maliciosos e as cores frias, comportamentos normais. Foi identificado que as imagens geradas pertencentes a mesma família possuem um mesmo padrão, permitindo realizar a classificação a partir delas, como feito, onde se atingiu acurácia entre 95,92% e 98,92%, indicando assim, que o uso de imagem do comportamento de malware pode produzir uma alta taxa de acerto de classificação.

Complementando a pesquisa já mencionada na seção anterior (Han et al., 2013), em Han et al. (2015) os autores propuseram um método para classificação convertendo arquivos binários em imagens e gráficos de entropia. O arquivo é convertido em imagem bitmap, desse ponto é calculado o valor de entropia de cada linha, gerando o gráfico de entropia, que é comparado com outros através de um algoritmo de cálculo de similaridade de gráfico de entropia, atingindo uma acurácia de 97,9%. Quando um arquivo binário é empacotado, as seções possuem alto valor de entropia, isso acaba sendo uma desvantagem para o método, porém os autores afirmam que pode ajudar a reduzir a quantidade de arquivos que precisam passar pela análise estática ou dinâmica e o método consegue distinguir famílias de *malware*.

Ainda apostando na visualização, Zhang et al. (2016) apresentam uma abordagem baseada em sequências *opcode*, onde cada pixel recebe o valor correspondente a multiplicação da probabilidade da sequência *opcode* e seu ganho de informação, e então, para extraír as características da imagem e reduzir o espaço de busca, é usado PCA (Principal Component Analysis). Para classificar é usado o método KNN, com k = 1, atingindo acurácia entre 94,8 e 96,5%, melhor que as outras abordagens avaliadas.

Para comparações, um *survey* de análise e classificação de *malware* é apresentado por Gandotra et al. (2014). Os autores focam na análise estática e dinâmica que são as mais utilizadas

3.3 APLICAÇÕES DE TEXTURA

A análise de textura é um recurso utilizado em muitas áreas da computação e vem mostrando ótimos resultados. Bertolini et al. (2013) fazem a verificação e identificação de escritores usando descritores de textura, produzindo melhores resultados do que os já obtidos anteriormente na literatura. Costa et al. (2012) classificam gêneros musicais extraindo recursos de textura dos espectogramas do sinal de áudio, onde concluem que os recursos de textura sempre superam os recursos baseados em conteúdo de áudio.

Thakare et al. (2013) apresentam um *survey* com diferentes técnicas de classificação de texturas e Nanni et al. (2012) também exibem técnicas de classificação de imagens, mas usando descritor de texturas LBP.

Em Conti et al. (2008), Conti et al. (2010a) e Conti et al. (2010b) os autores procuram abordagens visuais para identificar arquivos e estruturas de arquivos desconhecidas. Apresentam um classificador para identificação de tipos de fragmentos binários primitivos e fazem uma análise visual de objetos de dados binários como representações gráficas em escala de cinza. Em Conti et al. (2008) é implementado um sistema de engenharia reversa visual baseado nessa análise, onde concluem que as abordagens visuais ajudam a identificar rapidamente arquivos e analisar estruturas desconhecidas. Os autores ainda identificam que o futuro da análise visual de dados binários seria promissor. Foi realizada a análise visual de objetos de dados binários como representações gráficas em escala de cinza por Conti et al. (2010a) onde entendem que ela ajuda a distinguir regiões de dados estruturalmente diferentes, facilitando tarefas analíticas. Nesse artigo os autores oferecem uma taxonomia descritiva de fragmentos binários primitivos e suas representações gráficas. Já em Conti et al. (2010b), estudam meios automatizados para mapear grandes objetos binários e testam a classificação de diferentes tipos de arquivos apenas pela estrutura bruta dos dados, através da análise de textura. Neste artigo os autores sugerem que essa forma de analisar, pode auxiliar a tarefa de análise de *malware*.

A partir disso, nas pesquisas de Nataraj (2015), Nataraj et al. (2011a), Nataraj et al. (2013) e Nataraj et al. (2011b), os autores consideram, então, esse novo tipo de análise, diferente das propostas de análise dinâmica e estática, apresentando a análise de textura de *malware*. Com a estratégia baseada em textura binária podem produzir resultados de classificação próximos aos de análise dinâmica, mas a 1/4000 quantidade de tempo por binário (Nataraj et al., 2011b). Os arquivos são visualizados em imagens de escala de cinza, onde as imagens pertencentes à mesma família parecem muito semelhantes em layout e textura (Nataraj et al., 2011a). De acordo com Nataraj (2015), eles tratam o malware como imagens 2D em escala de cinza e utilizam de descritores de similaridade de imagem para caracterizá-los.

Os resultados experimentais preliminares já em 2011 eram bastante promissores com precisão de classificação de 98% (Nataraj et al., 2011a). Em Nataraj et al. (2011b) afirmam que obtiveram mais de 97% dos dados com rotulagem consistente. Em Nataraj et al. (2013) é explicado sobre o SARVAM, Sistema de Pesquisa e Recuperação de Malware baseado em

conteúdo, onde para a pesquisa e recuperação serem rápidas, foi utilizada uma técnica escalável de busca de vizinho mais próximo baseada em Balltree. Nataraj (2015) apresenta métodos ortogônicos não só pelo processamento de imagem, como por processamento de sinal, onde trata o *malware* como sinal 1D. O autor ainda afirma que o uso de métodos novos e complementares tornaria mais difícil e/ou caro para que os atacantes escapem de todos os sistemas de detecção. A partir dos estudos realizados por esses autores, pesquisas adicionais sobre classificação de *malware* baseada em textura foram realizadas e são o foco deste trabalho.

3.4 REVISÃO SISTEMÁTICA

Para selecionar os trabalhos que foram usados para avaliação, primeiramente foi feita uma revisão sistemática baseada no modelo proposto por Sampaio e Mancini (2007) que segue o modelo exibido na Figura 3.1. Esta revisão foi realizada com o objetivo final de avaliar as atuais técnicas utilizadas na literatura que são baseadas em análise de texturas para classificação de famílias de *malware*.

A busca foi feita utilizando bases de dados eletrônicas, como *Google Scholar* e *Scopus*, as quais foram consultadas com filtros a partir do ano de 2009 usando as seguintes palavras-chave: *malware classification* AND *texture analysis* AND *malware images* AND *visualization*. Todos os artigos avaliados foram publicados entre janeiro de 2009 e julho de 2018, pois, apesar de Nataraj et al. (2011a) afirmarem ter proposto uma abordagem nova e completamente diferente de analisar *malware*, estendeu-se a busca a dois anos anteriores para verificar se não havia artigos com a mesma proposta e para verificar os últimos 10 anos de pesquisas sobre o assunto.

Optou-se pela avaliação apenas de trabalhos publicados em inglês e com acesso gratuito ou disponibilizado pela Universidade Federal do Paraná (UFPR). Na Figura 3.2 é possível notar um aumento no número de publicações usando análise por texturas ano a ano, enquanto na Figura 3.3 observa-se o crescimento na quantidade de diferentes autores que publicaram ano a ano, indicando que o interesse por esse tipo de análise tem aumentado e esta pode ser uma boa estratégia para classificação.

Foram selecionadas apenas pesquisas de caráter experimental quanto a classificação de *malware* em famílias, restringindo-se àquelas cuja análise da amostra é feita através da conversão do binário para texturas em escala de cinzas. Ainda foram considerados apenas os experimentos tendo como desfecho a acurácia da classificação e com amostras de no mínimo cinco mil exemplares de *malware*.

Não foram utilizados experimentos com objetivo limitado à detecção ou análise de *malware* e, visando a qualidade metodológica dos estudos, foram excluídos artigos que traziam informações repetidas.

A pesquisa inicial identificou 573 publicações, porém, após a análise e seleção de acordo com as especificações mencionadas anteriormente, restaram 17 que foram aqui avaliadas. Os trabalhos selecionados são exibidos na Tabela 3.1. É possível visualizar que a maioria das pesquisas é realizada na Índia, seguida dos Estados Unidos da América (EUA), Brasil, Austrália, Japão, Coréia do Sul e Grécia.

3.4.1 Técnicas de Classificação

Nataraj et al. (2011a) propuseram pela primeira vez o uso de análise de textura para classificação de *malware*. Em sua pesquisa os autores mostram, através de alguns experimentos, que ao converter um binário de *malware* em uma textura em escala de cinzas é possível identificar padrões em amostras pertencentes à mesma família. Utilizando o descritor global de imagens

Tabela 3.1: Revisão Sistemática de Classificação de *Malware* Baseada em Textura.

Autores	**Ano**	**Título**	**País**
Nataraj et al.	2011	*Malware images: Visualization and automatic classification.*	EUA
Nataraj et al.	2011	*A comparative assessment of malware classification using binary texture analysis and dynamic analysis.*	EUA
Makandar and Patrot	2015	*Malware analysis and classification using artificial neural network.*	Índia
Makandar and Patrot	2015	*Malware image analysis and classification using support vector machine.*	Índia
Nataraj	2015	*A signal processing approach to malware analysis.*	EUA
Makandar and Patrot	2016	*An approach to analysis of malware using supervised learning classification.*	Índia
Kosmidis and Kalloniatis	2017	*Machine learning and images for malware detection and classification.*	Grécia
Yue	2017	*Imbalanced malware images classification: a CNN based approach.*	EUA
Luo and Lo	2017	*Binary malware image classification using machine learning with local binary pattern.*	EUA
Makandar and Patrot	2017	*Malware class recognition using image processing techniques.*	Índia
Makandar and Patrot	2017	*Wavelet statistical feature based malware class recognition and classification using supervised learning classifier.*	Índia
Singh	2017	*Malware classification using image representation.*	Índia
Rezende et al.	2017	*Malicious software classification using transfer learning of resnet-50 deep neural network.*	Brasil, Austrália
Rezende et al.	2018	*Malicious software classification using vgg16 deep neural network's bottleneck features.*	Brasil, Austrália
Yakura et al.	2018	*Malware analysis of imaged binary samples by convolutional neural network with attention mechanism.*	Japão
Makandar and Patrot	2018	*Trojan malware image pattern classification.*	Índia
Kabanga and Kim	2018	*Malware images classification using convolutional neural network.*	Coréia do Sul

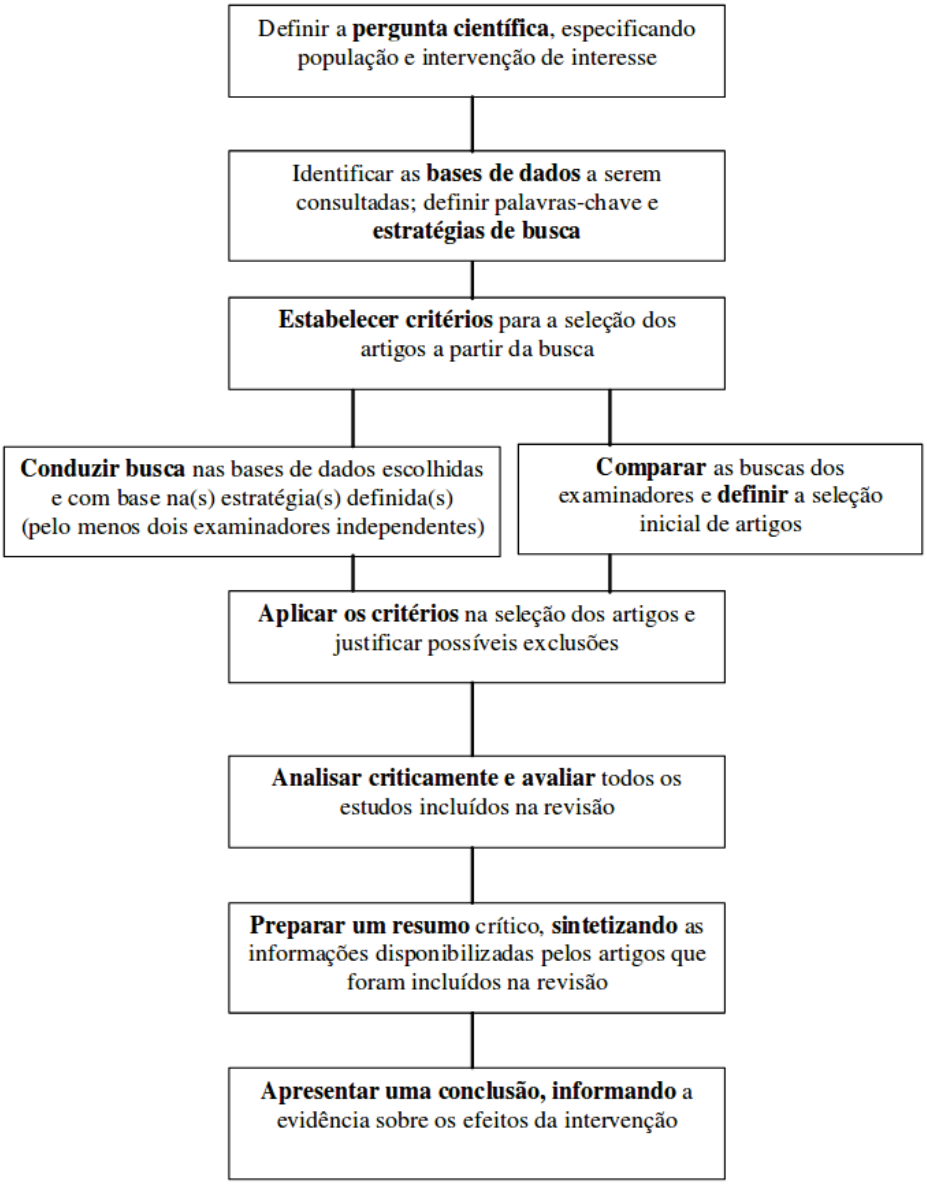


Figura 3.1: Processo de revisão sistemática da literatura. Fonte: Sampaio e Mancini (2007).

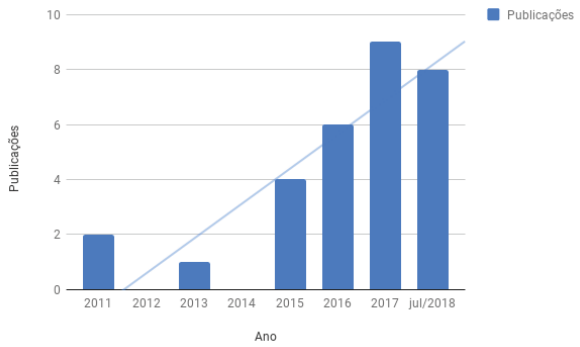


Figura 3.2: Número de publicações que utilizam análise de texturas de *malware* por ano.

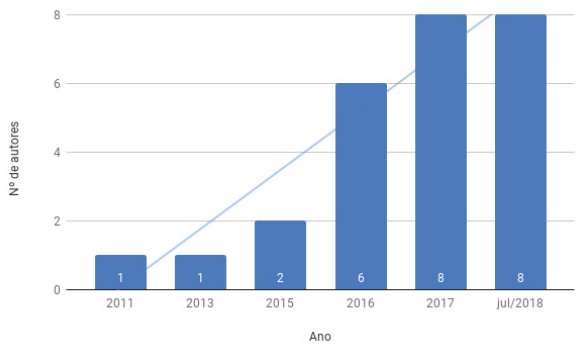


Figura 3.3: Número de autores diferentes com publicações que utilizam análise de texturas de *malware* por ano.

GIST e o classificador k-nearest neighbors com distância euclidiana, atingem 98% de acurácia. Os autores afirmam que o uso desse tipo de análise é bastante vantajoso pela velocidade e por ser agnóstico a plataforma, enquanto os outros tipos de análise de *malware* normalmente se restringem a um único sistema operacional. Porém possui as desvantagens que, realocando seções no binário ou adicionando uma grande quantidade de dados redundante, podem resultar em falhas de classificação.

Em Nataraj et al. (2011b), os autores realizam uma comparação direta da análise de texturas com a análise dinâmica, usando as bases Host-Rx, Malhuer e VX Heavens. Os autores mostram que a análise de textura atinge resultados comparáveis, porém 4000 vezes mais rápidos e, ainda, essa abordagem se mostrou resiliente à técnicas de ofuscação comuns da análise estática.

Seguindo a mesma estratégia de análise de textura, Makandar e Patrot (2015a) também realizam a classificação de *malware*. Usando descritor GIST e Transformada Wavelet de Gabor (GWT) e classificando com redes neurais artificiais obtiveram acurácia de 96,35% no *dataset* Malheur. E em Makandar e Patrot (2015b), classificando com SVM (Support Vector Machine), atingiram 89,68% no mesmo conjunto de dados.

Em sua tese, Nataraj (2015) propôs o uso de análise de textura e de sinais, afirmando que utilizar diferentes métodos de classificação e complementares entre si, torna-se mais difícil e/ou caro produzir tantas técnicas de evasão para criadores de *malware*. Para classificação usando análise de textura foi usado classificador KNN com acurácia de 97,4%, 98,37%, 83,27% e 84,55% nos conjuntos de dados Malimg (Windows) Malheur (Windows) VxShare (Linux) e Malgenome (Android), respectivamente.

Diferentes classificadores foram utilizados em Kosmidis e Kalloniatis (2017), onde os autores continuaram usando o descritor GIST porém avaliaram diferentes algoritmos de classificação usando *Machine Learning*. Os classificadores DT, *Perceptron*, MLP, SGD, RF e NC atingiram acurácia acima de 87% no *dataset* Malimg (Nataraj et al., 2011a), tendo como melhor classificador, nesse caso, o *Random Forest* com 91,6% de acurácia.

Em Yue (2017) experimenta-se o classificador CNN nas texturas de *malware* do *dataset* Malimg. Os autores obtiveram resultados de classificação bastante relevantes com acurácia de 98,63% no *dataset*.

Uma comparação entre o uso do descritor global GIST e o local LBP é realizada em Luo e Lo (2017). Os autores utilizam a análise de textura diferenciando o tipo de descritor e aplicando aos classificadores CNN com a biblioteca TensorFlow, SVM e KNN. Para o LBP, a acurácia apresentada por eles foi de 93,17%, 87,88% e 85,93%, já usando GIST 87,88%, 81,23% e 82,83%, mostrando assim que o uso do descritor local ajuda a produzir melhores resultados para todos os classificadores.

Makandar e Patrot (2017a) ainda usam análise de imagens baseada nas características globais. Para calcular as similaridades usam Transformada Wavelet Discreta (DWT) e a classificação é feita usando classificadores KNN e SVM. Alcançaram uma acurácia de 98,8% no *dataset* Malimg usando SVM e 98,84% usando KNN. Alterando o redimensionamento inicial das imagens e usando SVM, Makandar e Patrot (2017b) atingiu 92,53% no *dataset* Malimg e 91,05% no Malheur.

Redes residuais com 152 camadas foram utilizadas em Singh (2017) alcançando 98,21% em um *dataset* grande criado pelos autores e 96,08% de acurácia no *dataset* Malimg. Também utilizando redes residuais, porém com 50 camadas e diferente redimensionamento, pré-treinadas no *dataset* ImageNet, Rezende et al. (2017) alcançou 98,62% de acurácia no *dataset* Malimg.

Em Rezende et al. (2018), os autores usaram CNN com a arquitetura *Visual Geometry Group* com 16 camadas (VGG16) pré-treinado no *dataset* ImageNet para extração de características, classificando com SVM o *dataset* VirusSign, obtendo 92,97% de acurácia. Já em Makandar

e Patrot (2018), os autores também usam SVM para classificação, além de KNN, porém extraindo características com GWT, o que alcançam, num subconjunto do *dataset* Malimg, 75,11% de acurácia com SVM e 89,11% com KNN.

Yakura et al. (2018a) fazem classificação de texturas de *malware* utilizando o mesmo método proposto por Nataraj et al. (2011a) e a textura gerada é aplicada no CNN, utilizando o conjunto de dados VX Heavens. Aplicaram o método proposto em Nataraj et al. (2011b) no mesmo *dataset* também para comparação e atingiram 53,53% no método tradicional e 50,97% no proposto usando CNN (taxa de erro Top-1), e 41,78% e 31,34% para taxa de erro Top-5.

Kabanga e Kim (2017) utilizam o *dataset* Malimg e fazem uso de CNN para classificação, afirmando atingir assim uma acurária de 98%. Os autores criticam o método de análise de textura por afirmarem que tendo uma pequena alteração na imagem, mesmo que não visível aos olhos humanos, pode resultar em uma classificação errada.

3.4.2 Categorização da Literatura

Um classificador recebe como entrada algumas características do exemplar a ser avaliado, normalmente extraídas por um descritor de texturas. O objetivo é determinar a qual família uma amostra pertence, tornando a escolha do descritor de suma importância para a classificação.

A Tabela 3.3 mostra que o descritor mais comumente utilizado é o GIST. Nataraj et al. (2011a), que propuseram pela primeira vez a utilização da análise de texturas para classificação de *malware*, usavam o descritor de texturas GIST como entrada do classificador, o qual apresentou resultado melhor do que outros descritores globais (HTD e CLD), e que também foi o descritor utilizado em outros trabalhos (Nataraj et al., 2011b) (Nataraj, 2015) (Kosmidis e Kalloniatis, 2017) (Luo e Lo, 2017). Seguindo a mesma ideia, Makandar e Patrot (2015a) utilizam como descritor o GWT em conjunto com GIST, pois a escolha do descritor de textura é de fundamental importância para a correta classificação. Já Makandar e Patrot (2015b) e Makandar e Patrot (2018) utilizam GWT como único descritor, porém não atingem uma acurácia de classificação tão alta. Makandar e Patrot (2016) propõem a utilização do DWT em conjunto com GIST, aumentando a taxa de acerto, enquanto que o DWT é utilizado sozinho por Makandar e Patrot (2017a) e Makandar e Patrot (2017b). Luo e Lo (2017) decidem utilizar um descritor local para evitar erro de classificação nos casos de realocação de seções ou quando se é adicionado grande quantidade de dados redundantes, problema já sugerido em Nataraj et al. (2011b), mostrando que mesmo com diferentes classificadores, o descritor local apresenta melhores acurácia. Cabe ressaltar que algoritmos baseados em *deep learning* não necessitam de descritor da textura, pois o próprio classificador seleciona as características da imagem mais importantes.

A análise dos trabalhos da área deixa claro que o uso de diferentes descritores interfere na taxa de acerto da classificação. É possível perceber isso comparando os resultados de Nataraj et al. (2011a) e Makandar e Patrot (2017a), nos quais os autores fazem uso do mesmo classificador (KNN) e conjunto de dados, porém alcançam taxas de acurácia distintas, pois se diferenciam no descritor. Isso pode ser também constatado na comparação feita por Luo e Lo (2017) que avalia dois descritores em diferentes classificadores, mostrando que ao alterar o descritor, tem-se acurácia diferente em todos os classificadores testados. Apesar de ser difícil realizar uma comparação direta entre todos os trabalhos (diferentes *datasets* e redimensionamentos), a Figura 3.4 mostra a acurácia alcançada por classificadores que utilizam descritor GIST, enquanto que a Figura 3.5 mostra a acurácia obtida com outros descritores.

Os classificadores cujo descritor é o GIST utilizaram o mesmo *dataset* em todos os casos (Figura 3.4), porém há diferenças quanto ao redimensionamento da imagem, o que pode ter interferido nos resultados. Já a classificação apresentada na Figura 3.5 referente aos demais

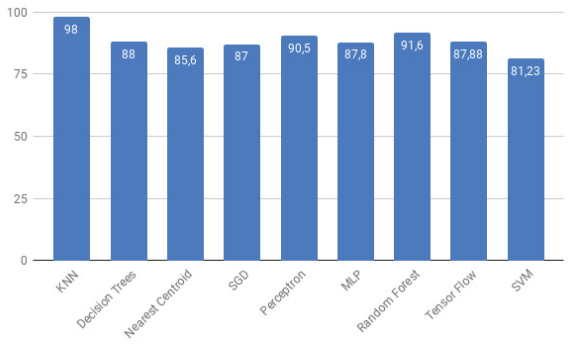


Figura 3.4: Acurácia de classificação utilizando descritor GIST em diferentes classificadores.

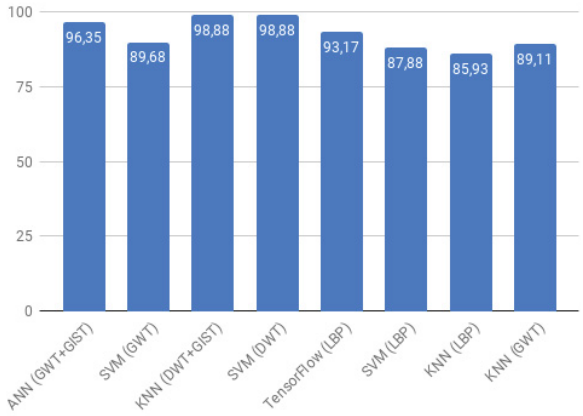


Figura 3.5: Acurácia de classificação utilizando outros descritores em diferentes classificadores.

descritores exibe resultados um pouco mais difíceis de comparar, pois os autores fazem uso de *datasets* distintos em alguns casos, além dos diferentes redimensionamentos.

Mesmo assim, observa-se que o KNN apresenta as melhores taxas de acerto com diferentes descritores, porém essa acurácia diminui quando usado o LBP e o GWT, o que mostra que o classificador não é a única variável importante para classificação de *malware* por análise de texturas. O SVM, exceto em um trabalho com um *dataset* reduzido, apresentou as taxas de acerto mais baixas da literatura, podendo significar que para esses conjuntos de dados, não consegue fazer a classificação tão bem. O RF apresentou a segunda melhor taxa de acerto com o descritor GIST e não foi utilizado com outros descritores, sendo um classificador que poderia ser mais explorado.

Ainda, como já mencionado, alguns autores não utilizam descritor de texturas, pois o próprio classificador faz a seleção das características. Na Figura 3.6, exibe-se a acurácia das redes neurais convolucionais propostas na literatura conforme sua arquitetura e para *datasets* distintos. As CNNs com mais camadas apresentaram pouco melhores taxas de acerto, porém problemas de degradação e desaparecimento de gradientes costumam aparecer conforme aumenta-se a profundidade da rede neural (Singh, 2017), portanto é necessário utiliza-las com cautela. Novamente há uma dificuldade em comparar os resultados devido ao redimensionamento inicial da imagem.

Outro fator que pode influenciar na classificação é o tamanho da imagem de entrada. Como visto, na etapa de pré-processamento da imagem, alguns autores fazem o redimensionamento da mesma, deixando-a quadrada. Nataraj et al. (2013) definem esse valor como 64x64, pois um valor menor não retorna uma assinatura significativa e um maior aumenta a complexidade computacional. Porém, esse valor é escolhido empiricamente e não há um consenso sobre o melhor valor a ser fixado. A Figura 3.7 mostra a diferença da pior e melhor acurácia obtida na classificação por texturas com redimensionamento de imagem, desconsiderando-se descritor e classificador utilizados.

Levando-se em conta só os resultados obtidos com o redimensionamento da imagem, apesar da melhor acurácia ser o de 64x64, talvez o mais prudente seria utilizar um redimensionamento por 224x224, dada a menor variação e mantendo alta taxa de acerto em todos os casos, porém, como é um valor atribuído empiricamente, é necessário avaliá-los para identificar o melhor valor.

A utilização de diferentes *datasets* dificulta a comparação real das técnicas apresentadas. Foram identificados oito *datasets* utilizados na literatura, mas o mais usado é o Malimg (Nataraj

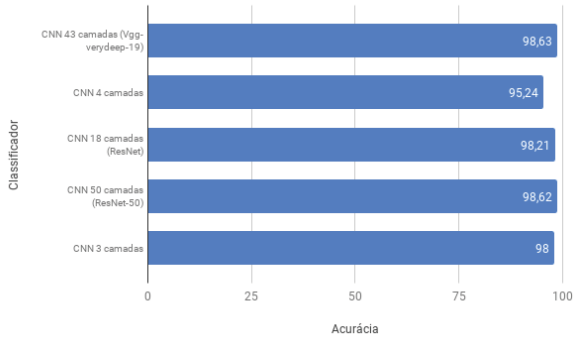


Figura 3.6: Acurácia obtida para diferentes arquiteturas de CNN.

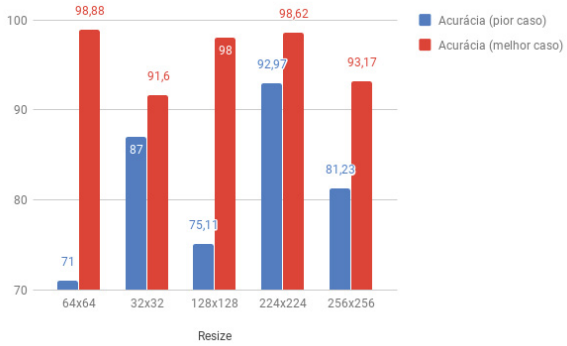


Figura 3.7: Melhor e pior acurácia obtidas em cada redimensionamento.

et al., 2011a), que possui 9342 amostras de 25 famílias de *malware* já convertidas em textura. A Tabela 3.2 mostra as informações básicas de cada *dataset* conforme a literatura e a acurácia obtida pelos autores em cada base.

Tabela 3.2: *Datasets* utilizados na literatura para classificação de *malware* em famílias.

Nome do *Dataset*	**Exemplares**	**Famílias**	**Acurácia** (%)
Host-Rx	393	6	95.14
Malheur	3131	24	89.68 a 98.37
VX Heaven	63002	531	72.80
Malimg	9342	25	75.11 a 98.88
MalGenome	1094	13	97.40
VXShare	568	8	83.27
Singh 2017 (VirusShare, VirusTotal, Malshare)	44945	20	95.24 a 98.21
VirusSign	10136	20	92.97

Nota-se na Tabela 3.2 que, para um *dataset* com uma grande quantidade de famílias, a taxa de acerto classificação diminui. Porém, como em outros casos, o classificador e descritor utilizados podem também ter interferido nesse resultado. Para uma comparação justa, seria importante usar a mesma técnica para todos os *datasets*.

Outro fator importante que interfere diretamente nos resultados é a escolha do classificador. Nataraj et al. (2011a), Nataraj et al. (2011b), Nataraj (2015), Makandar e Patrot (2016), Luo e Lo (2017), Makandar e Patrot (2017a) e Makandar e Patrot (2018) optam pelo KNN que, apesar de ser um classificador bastante simples (assume que todas as instâncias correspondem a pontos no espaço e relaciona-as com suas vizinhas (Mitchell, 1997)), apresenta uma boa taxa de acurácia. KNN atingiu o melhor resultado dentre os demais classificadores utilizados para texturas de *malware* (Makandar e Patrot, 2016). Makandar e Patrot (2015b), Luo e Lo (2017), Makandar e Patrot (2017a), Makandar e Patrot (2017b) e Makandar e Patrot (2018) também utilizam o classificador SVM, que é efetivo em espaços de grande dimensão e apresentou resultados bastante significativos, alcançando a melhor acurácia dentre as pesquisas apresentadas em Makandar e Patrot (2017a), o mesmo alcançado anteriormente utilizando KNN. Kosmidis e Kalloniatis (2017) fazem uma comparação entre diferentes classificadores, apresentando não só a acurácia de classificação, mas o tempo médio de treino e teste de cada um deles. Redes Neurais Artificiais e suas variações foram apresentadas em Makandar e Patrot (2015a), Singh (2017), Rezende et al. (2017), Yakura et al. (2018b) e Kabanga e Kim (2017), mostrando que as pesquisas mais recentes têm apostado em sua utilização para classificação de *malware*, pois apresentam altas

taxas de acurácia e por vezes são mais rápidas que outras abordagens, como o uso de CNN com a biblioteca *TensorFlow* (Luo e Lo, 2017).

A Tabela 3.3 exibe o estado da arte em classificação de *malware* por análise de texturas, diferenciando os trabalhos por descritor, classificador, redimensionamento (*resize*), acurácia alcançada por cada técnica e *dataset* utilizado. Com essas informações extraídas dos artigos selecionados, é possível identificar as técnicas mais aplicadas e que têm apresentado melhor resultado. Na próxima seção é feita uma comparação para categorizar esses artigos.

Tabela 3.3: Comparação de técnicas usadas para classificação de *malware* usando análise de texturas (*n/i*: não informado pelo autor; ✗: descritor não necessário; [1]: Taxa de erro Top 5).

Referência	***Resize***	**Descritor**	**Classificador**	**Acurácia (%)**	***Dataset***
Nataraj et al. (2011a)	*n/i*	GIST	KNN	98,00	Malimg
Nataraj et al. (2011b)	64x64	GIST	KNN	95,14 97,57 72,80	Host-Rx Malheur VX Heaven
Makandar e Patrot (2015a)	64x64	GWT + GIST	ANN	96,35	Malheur
Makandar e Patrot (2015b)	*n/i*	GWT	SVM	89,68	Malheur
Nataraj (2015)	64x64	GIST	KNN	97,40 98,37 97,40 83,27	Malimg Malheur MalGenome VXShare
Makandar e Patrot (2016)	*n/i*	DWT, GIST	KNN	98,88	Malimg
Kosmidis e Kalloniatis (2017)	32x32	GIST	DT NC SGD Perceptron MLP RF	88,00 85,60 87,00 90,50 87,80 91,60	Malimg
Yue (2017)	*n/i*	✗	CNN	98,63	Malimg
Luo e Lo (2017)	*n/i*	LBP GIST	CNN SVM KNN CNN SVM KNN	93,17 87,88 85,93 87,88 81,23 82,83	Malimg
Makandar e Patrot (2017a)	64x64	DWT	KNN SVM	98,84 98,88	Malimg
Makandar e Patrot (2017b)	256x256	DWT	SVM	91,05 92,53	Malheur Malimg
Singh (2017)	32x32	✗	CNN CNN (ResNet) CNN (ResNet)	95,24 98,21 96,08	Malshare + VirusShare + VirusTotal. Malimg
Rezende et al. (2017)	224x224	✗	CNN (ResNet)	98,62	Malimg
Rezende et al. (2018)	224x224	✗	SVM	92,97	VirusSign
Yakura et al. (2018b)	64x64	✗	CNN	31,34[1]	VX Heaven
Makandar e Patrot (2018)	128x128	GWT	KNN SVM	89,11 75,11	Malimg
Kabanga e Kim (2017)	128x128	✗	CNN	98,00	Malimg

3.5 CONCLUSÃO

Neste capítulo observou-se diversas pesquisas sobre análise e classificação de *malware*, usando diferentes técnicas e abordagens, todas com vantagens e desvantagens, apontando diversas estratégias para combatê-las.

Apresentou-se também outras aplicações que fazem o uso de textura e visualização, que mostraram que a estratégia da análise de textura pode ser uma grande aliada para obtenção de melhores resultados de classificação de *malware*.

Por fim foi apresentada uma revisão sistemática da literatura realizada para avaliação de técnicas de classificação baseadas na análise de textura de *malware*. Foram exibidas comparações entre as técnicas e as dificuldades encontradas, pois não é possível compará-las diretamente.

No próximo capítulo serão apresentados os passos seguidos para prosseguir a avaliação do uso da análise de textura para a classificação de *malware* e os *datasets* utilizados.

4 DESENVOLVIMENTO E METODOLOGIA

Neste capítulo é apresentado o desenvolvimento do trabalho. São exibidos os passos que devem ser seguidos para classificação de *malware* com uso de análise de textura, as informações sobre a implementação e configuração dos experimentos e, por fim, são apresentados os *datasets* utilizados para comparação.

4.1 METODOLOGIA DA CLASSIFICAÇÃO

Para realizar a classificação de *malware* por análise de texturas foi utilizada a mesma metodologia de Nataraj et al. (2011a), disponível em Laks (2014), na qual são realizados cinco passos descritos a seguir e ilustrados na Figura 4.1.

Passo 1 - Converter binários em imagens digitais: cada byte do arquivo binário é representado por um píxel na escala de cinzas (0, preto; 255, branco), preenchidos após definir uma largura fixa (Nataraj, 2015), enquanto a altura varia conforme o tamanho do arquivo. Essa conversão é bastante rápida, no desktop usado neste trabalho, por exemplo, para converter os binários do *dataset* local levou cerca de 0.0163 segundos por amostra.

Passo 2 - Organizar o *dataset*: para posterior classificação, as amostras são separadas de acordo com sua família. As amostras são categorizada em famílias de acordo com os rótulos obtidos pelo VirusTotal (2017). Para o *dataset* local foi utilizado o AVClass (Sebastián et al., 2016) para seleção do rótulo.

Passo 3 - Calcular vetor de características: são extraídas as características das amostras, pelas quais será calculada a semelhança entre as variantes das famílias. Para computar as características, é comum usar operações de redimensionamento, filtragem e média de blocos (Nataraj, 2015). Neste trabalho foram utilizados o descritor GIST e o LBP para obter as características com um redimensionamento de 128x128, valor atribuido empiricamente que apresentou melhores taxas de acurácia em experimentos preliminares, assim como o redimensionamento já utilizado na literatura.

Passo 4 - Classificação supervisionada: Tendo então o vetor de características é possível realizar a classificação. Para esse trabalho foram usados os algoritmos de classificação mencionados na Seção 2.4 que utilizam esse vetor de características de cada amostra para agrupar cada uma delas na família correspondente, ou seja, na família que possui características mais similares às características encontradas nessa amostra.

Passo 5 - Visualização de resultados: utiliza-se algumas métricas de classificação. Optou-se pela acurácia para comparar a taxa de acerto entre os classificadores e a matriz de confusão, para se observar o resultado obtido para cada família.

4.2 IMPLEMENTAÇÃO

Os experimentos foram feitos com validação cruzada 10-*fold*, pré-processamento das imagens redimensionando conforme a literatura e para 128x128 e extração de características com

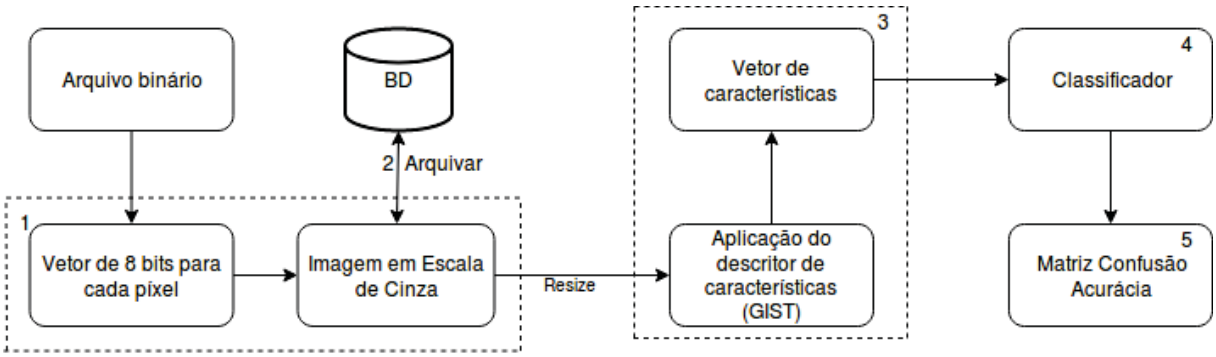


Figura 4.1: Passos da classificação de *malware* baseada em análise de texturas.

um descritor global (GIST) e um local (LBP) para comparação. Para o descritor global foram utilizados 20 filtros, com uma dimensionalidade de 320 através da biblioteca `pyleargist`. Para o descritor local foi utilizada o *toolkit* `scikit-image` (van der Walt et al., 2014), subdividindo a imagem em blocos 3x3, avaliando os 8 vizinhos e usando o método *uniform*, que é invariante a rotação e escala de cinzas. Os algoritmos de classificação foram usados com os mesmos parâmetros indicados pela literatura ou manteve-se os valores *default* do `scikit-learn`.

Para classificação foram importados do pacote `sklearn` os seguintes algoritmos: `neighbors.KNeighborsClassifier` para o classificador KNN, usando 3 vizinhos; `tree.DecisionTreeClassifier` para o DT com as configurações *default*; `ensemble.RandomForestClassifier` para RF também com valores *default* (10 estimadores); `neighbors.nearest_centroid.NearestCentroid` para o NC com valores *default*; `svm.LinearSVC` para o SVM também com valores *default*; `linear_model.SGDClassifier` para o SGD com os parâmetros *loss="hinge", penalty="l2"* e para o Perceptron com os valores *loss='perceptron', eta0=1, learning_rate='constant', penalty=None* (Kosmidis, 2017); `neural_network.MLPClassifier` para o MLP com valores *default*.

Para o CNN foi usada a biblioteca de redes neurais `Keras` usando os mesmos valores, usando ResNet-50 e treinando previamente no *dataset* ImageNet, com os mesmos parâmetros usado em (Rezende et al., 2017). As texturas em escala de cinzas foram convertidas em imagens RGB, redimensionadas em 224x224 e foi reduzido de cada pixel o valor RGB médio computado no ImageNet. Para o ResNet, foi reduzida a última camada *softmax* totalmente conectada de 1000 para o mesmo número de famílias do *dataset* correspondente, usou-se o otimizador Adam e 2000 *epochs*.

Os experimentos foram realizados em um computador Desktop Ubuntu 16.04 LTS com processador Intel Core i5-650, 3.2GHz de frequência 4MB de cache, 2 núcleos e 4 threads, além de 4GB de memória RAM. Para rodar as redes neurais convolucionais foi usado um servidor Ubuntu 14.04.3 LTS Intel Xeon E5620 de 2.4GHz de frequência, 12MB de cache, 4 núcleos e 8 threads e com 32GB de memória RAM.

As métricas para avaliação dos classificadores selecionadas foram implementadas usando o módulo `sklearn.metrics`. Foi optado pelo uso da acurácia, que é a métrica mais comumente utilizada para classificação pela literatura e a matriz confusão para melhor visualização do resultado.

A acurácia mede a taxa de acerto de uma classificação, para os experimentos realizados nessa pesquisa, foi utilizada a função `accuracy_score`. Corresponde às amostras classificadas exatamente na família correspondente.

Para visualizar melhor a acurácia é utilizada a matriz confusão. Ela é uma matriz com representação dos valores reais pelos valores preditos, onde, em um caso ideal, a diagonal principal, que corresponde ao mesmo valor real e predito, seria a única parte da matriz preenchida. Para gerar a matriz confusão foi utilizada a função `confusion_matrix`.

A seleção dos dados de treinamento e teste foi realizada usando validação cruzada k-fold, um método estatístico usado avaliar modelos de aprendizado de máquina. A base de dados é dividida em k grupos aleatoriamente (neste caso 10), onde são utilizados para treinamento k-1 grupos, e o outro grupo é usado para teste. Esse procedimento é realizado k vezes, para que todos os grupos sejam o grupo de teste uma vez, e aí então é calculada a acurácia do modelo.

Para avaliar a classificação de *malware*, é necessário partir de alguns pressupostos de que o *dataset* é confiável, não foi manipulado, e assumir que a rotulação escolhida é correta. Portanto, para validar o método, foram utilizados dois *datasets* diferentes, um público, com poucas famílias, e um local, maior, com um cenário um pouco mais próximo da realidade, sem realizar seleção de amostras. Os *datasets* apresentaram resultados de classificação bastante distintos, o que levanta um questionamento sobre a qualidade dos dados usados no *dataset* público.

Para o *dataset* local procurou-se seguir as práticas recomendadas por Rossow et al. (2012), procurando não enviesar os resultados com a seleção de dados como mostrado por Li et al. (2010).

4.3 *DATASETS*

Para essa pesquisa foram utilizados dois *datasets*, um público e um local, afim de validar as técnicas de classificação já apresentadas pela literatura e obter um classificador mais genérico quanto ao conjunto de dados.

O *dataset* público tomado como referência foi o Malimg *Dataset* (Nataraj et al., 2011a), que possui 9342 amostras já convertidas em imagens em escala de cinza de 25 famílias de malware. Esse *dataset* foi utilizado em treze dos dezessete trabalhos encontrados na revisão sistemática. A Tabela 4.1 mostra a distribuição de malwares por família desse conjunto de dados. Observa-se pela tabela que existem muitas variações da mesma família consideradas como famílias diferentes.

Adicionalmente, utilizou-se um *dataset* local de *malware* capturados desde 2007, atingindo um total de 19719 amostras que, rotulando através do Sebastián et al. (2016) e VirusTotal (2017), foram agrupadas em 5815 famílias. Devido a grande quantidade de famílias, é exibido na Tabela 4.2 apenas um subconjunto da mesma, mantendo apenas as famílias mais significativas (pelo menos 50 amostras por família). Nos experimentos realizados nesse trabalho o *dataset* local completo foi utilizado e subconjuntos do mesmo, inclusive este apresentado na tabela.

4.4 CONCLUSÃO

Nesse capítulo foram explicados os passos para realizar a classificação de *malware* através da análise de textura, explicando as condições usadas nos experimentos realizados neste trabalho. Por último foram apresentados os *datasets* Malimg, proposto por Nataraj et al. (2011a), e local, cujas amostras foram obtidas principalmente da coleta de *malware* em atuação no cenário de Internet brasileiro.

No próximo capítulo serão exibidos os experimentos realizados conforme a metodologia aqui exibida nos dois *datasets*.

Tabela 4.1: Descrição do Malimg Dataset

	Classe	Família	Nº Variantes	Nº (%)
1	Worm	Allaple.L	1591	17,03
2	Worm	Allaple.A	2949	31,57
3	Worm	Yuner.A	800	08,56
4	PWS	Lolyda.AA 1	213	02,28
5	PWS	Lolyda.AA 2	184	01,97
6	PWS	Lolyda.AA 3	123	01,32
7	Trojan	C2Lop.P	146	01,56
8	Trojan	C2Lop.gen!G	200	02,14
9	Dialer	Instantaccess	431	04,61
10	Trojan Downloader	Swizzor.gen!I	132	01,41
11	Trojan Downloader	Swizzor.gen!E	128	01,37
12	Worm	VB.AT	408	04,37
13	Rogue	Fakerean	381	04,09
14	Trojan	Alueron.gen!J	198	02,12
15	Trojan	Malex.gen!J	136	01,46
16	PWS	Lolyda.AT	159	01,70
17	Dialer	Adialer.C	125	01,34
18	Trojan Downloader	Wintrim.BX	97	01,04
19	Dialer	Dialplatform.B	177	01,89
20	Trojan Downloader	Dontovo.A	162	01,73
21	Trojan Downloader	Obfuscator.AD	142	01,52
22	Backdoor	Agent.FYI	116	01,24
23	Worm:AutoIT	Autorun.K	106	01,13
24	Backdoor	Rbot!gen	158	01,69
25	Trojan	Skintrim.N	80	00,86

Tabela 4.2: Descrição do *dataset* local

	Classe	Família	Nº Variantes	Nº (%)
1	Backdoor	Agobot	202	02,86
2	Backdoor	Aimbot	73	01,03
3	Worm	Bagle	138	01,96
4	Trojan Banker	Banbra	75	01,06
5	Trojan	Bancos	327	04,63
6	Trojan Downloader	Banload	215	03,05
7	Backdoor	Bifrose	128	01,81
8	Trojan	Constructor	110	01,56
9	Trojan Downloader	Dadobra	145	02,05
10	Trojan	Delf	1267	17,96
11	Trojan Downloader	Dyfuca	84	01,19
12	Trojan	Goldun	58	00,82
13	Trojan	Harnig	58	00,82
14	Virus	Hllc	61	00,86
15	Virus	Hllo	60	00,85
16	Backdoor	Hupigon	332	04,71
17	Trojan Downloader	Inservice	115	01,63
18	Backdoor	Ircbot	151	02,14
19	Trojan Downloader	Istbar	199	02,82
20	Worm	Kelvir	82	01,16
21	Trojan PSW	Ldpinch	123	01,74
22	Trojan PSW	Lineage	101	01,43
23	Trojan	Lmir	410	05,81
24	Trojan	Lowzones	60	00,85
25	Worm	Mytob	86	01,22
26	Trojan	Nsanti	50	00,71
27	Backdoor	Pcclient	79	01,12
28	Trojan PSW	Qqpass	123	01,74
29	Trojan	Qqrob	58	00,82
30	Backdoor	Ranky	72	01,02
31	Backdoor	Rbot	909	12,88
32	Virus	Score	120	01,70
33	Backdoor	Sdbot	685	09,71
34	Worm	Spybot	122	01,73
35	Trojan Downloader	Swizzor	56	00,79
36	Backdoor	Wootbot	63	00,89
37	Trojan	Zlob	59	00,84

5 EXPERIMENTOS

Para reavaliar a viabilidade do uso de análise de texturas para classificação de *malware* em famílias, foram realizados diversos experimentos usando diferentes técnicas apresentadas na literatura em dois *datasets*. Neste capítulo serão apresentados os resultados obtidos pelos diferentes classificadores nos dois *datasets*, bem como a utilização de algumas técnicas de ofuscação bastante simples para testar sua eficiência diante de um binário ofuscado.

5.1 VALIDAÇÃO DOS CLASSIFICADORES EM DIFERENTES *DATASETS*

Nesta seção foram reimplementados os experimentos no *dataset* público (Subseção 5.1.1) e no local (Subseção 5.1.2). O *dataset* público foi avaliado completo e subdividido em 8 famílias (conforme empregado em Makandar e Patrot (2016) e Makandar e Patrot (2017a)). Já o *dataset* local foi avaliado de quatro maneiras: o *dataset* local completo; um subconjunto contendo apenas as famílias com mais de 50 amostras por famílias (pois um número maior de amostras em cada família caracteriza mais facilmente a mesma); um subconjunto das 10 maiores famílias, contendo no mínimo 100 amostras por família, usando 100 amostras por família e; um subconjunto das 8 maiores, também com apenas 100 amostras por família, pois há uma diferença muito grande na quantidade de amostras em cada uma delas e isso pode enviesar o resultado (Rossow et al., 2012).

5.1.1 Malimg Dataset

Nesta subseção serão exibidos os resultados de classificação para o *dataset* público completo e de um subconjunto de oito famílias do mesmo. As matrizes de confusão dos experimentos aqui realizados são encontradas no Apêndice A.

A Tabela 5.1 apresenta uma comparação da acurácia de classificação dos algoritmos aplicados no *dataset* completo. Foram utilizados um descritor global (GIST) e um local (LBP) para comparação e os dois descritores recebem as imagens redimensionadas na mesma escala apresentada pela literatura (32x32 ou 64x64) e a seguir em uma escala padronizada de 128x128 para poder comparar diretamente os classificadores.

Tabela 5.1: Porcentagem da acurácia de classificação no *dataset* Malimg de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128).

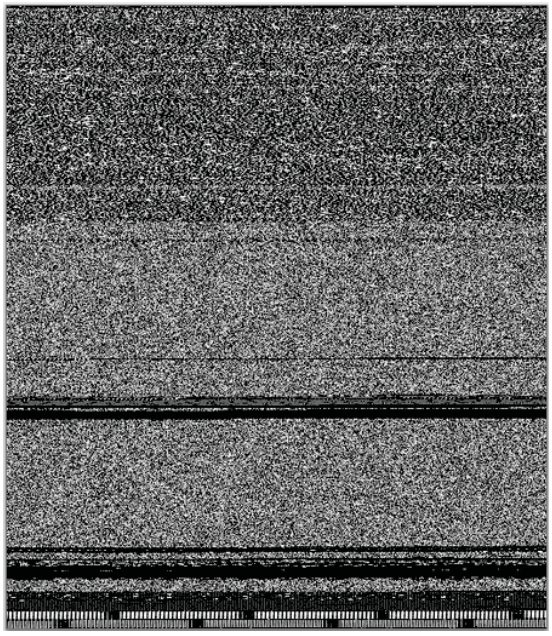
Classificador	**KNN**	**DT**	**RF**	**NC**	**SVM**	**SGD**	**Perceptron**	**MLP**
GIST[1]	96,97	93,06	95,23	91,33	95,23	91,44	90,03	95,99
LBP[1]	85,59	69,34	78,66	47,02	53,41	37,38	30,88	71,07
GIST[2]	97,94	96,32	97,83	96,42	95,88	94,58	91,33	96,42
LBP[2]	95,77	92,96	95,67	84,72	51,25	40,41	66,41	83,31

Usando o descritor GIST, neste *dataset* todos os classificadores apresentaram resultados acima de 90% de acurácia, já utilizando LBP, alguns classificadores tiveram um desempenho

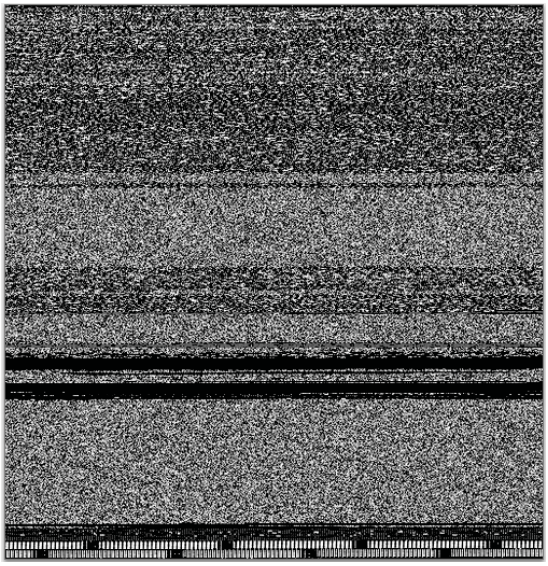
bem menor. Quando compara-se a acurácia de classificação quanto a escala, é possível notar que normalmente o redimensionamento padronizado de 128x128 apresenta melhores resultados e os tempos de treinamento e teste permanecem similares. Os tempos de treinamento e teste são exibidos na Tabela 5.3, no final da subseção.

Para o classificador CNN optou-se pela opção proposta por Rezende et al. (2017) que atingiu uma das mais altas taxas de acerto na literatura e com informações suficientes para replicação dos experimentos, usando arquitetura ResNet com 50 camadas pré-treinada no *dataset* ImageNet. No *dataset* público, atingiu-se 98,57% de acurácia com tempo médio de treinamento de 5589,2463 segundos e teste 0,4730 segundos.

Analisando os erros dos classificadores é observado uma confusão aceitável entre amostras pertencentes a variações da mesma família, por exemplo `Swizzor.gen!E` e `Swizzor.gen!I`. A Figura 5.1 mostra como arquivos dessas famílias podem ser realmente similares.



(a) `Swizzor.gen!E`



(b) `Swizzor.gen!I`

Figura 5.1: Variações das famílias `Swizzor.gen`.

Em Makandar e Patrot (2016) e Makandar e Patrot (2017a), os autores utilizam um subconjunto do *dataset* Malimg para realizar a classificação baseada em texturas. Usando as mesmas oito famílias utilizadas em Makandar e Patrot (2016) foram re-executados os experimentos, porém usando essas poucas famílias os classificadores conseguem atingir 100% de acurácia ou um valor muito próximo desse, como se pode ver na Tabela 5.2, mostrando que essa forma de abordar os dados pode enviesar os resultados por usar instâncias que são mais fáceis de classificar, similar ao estudo realizado por Li et al. (2010).

Novamente pela tabela observa-se que o uso do descritor GIST se sobressai perante ao LBP, assim como a escala padronizada diante da usada na literatura. O classificador CNN nesse subconjunto apresentou acurácia de 99,87% com duração média de treinamento de 1942,8040 segundos.

Uma das vantagens da utilização de análise de texturas para classificação de *malware* é ser mais rápida que as demais abordagens, portanto, outro ponto importante a ser analisado são os tempos de treinamento e teste de classificação. Os valores médios de tempo de treinamento e teste dos classificadores para o *dataset* público e seu subconjunto são exibidos na Tabela 5.3.

Tabela 5.2: Porcentagem da acurácia de classificação no subconjunto do *dataset* Malimg de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128; célula cinza com a maior acurácia e vermelha com a menor).

Classificador	**KNN**	**DT**	**RF**	**NC**	**SVM**	**SGD**	**Perceptron**	**MLP**
GIST[1]	100,00	99,81	99,81	97,91	99,81	99,43	91,63	99,81
LBP[1]	99,24	96,77	99,05	71,10	78,90	73,76	86,50	90,87
GIST[2]	100,00	99,81	100,00	99,05	100,00	100,00	100,00	100,00
LBP[2]	99,43	98,67	99,43	94,68	78,52	78,33	87,26	91,44

Tabela 5.3: Tempo médio em segundos de treinamento e teste por classificador no Malimg Dataset ([1]: *dataset* completo, [2]: subconjunto; célula cinza com o melhor tempo e vermelha com o pior).

Classificador	**Escala**	**Literatura**		**Padronizada**		**Literatura**		**Padronizada**	
	Descritor	**GIST**[1]	**LBP**[1]	**GIST**[1]	**LBP**[1]	**GIST**[2]	**LBP**[2]	**GIST**[2]	**LBP**[2]
KNN	**Treino**	00,1560	00,0096	00,1542	00,0111	00,0635	00,0065	00,0686	00,0066
	Teste	01,9433	00,0688	01,8197	00,0514	00,6251	00,0367	00,6705	00,0308
DT	**Treino**	05,1326	00,1127	04,6669	00,1371	00,6139	00,0425	00,4502	00,0483
	Teste	00,0009	00,0007	00,0009	00,0005	00,0005	00,0004	00,0005	00,0003
RF	**Treino**	01,4679	00,2012	01,7116	00,2047	00,2604	00,0994	00,1781	00,0794
	Teste	00,0051	00,0052	00,0051	00,0044	00,0022	00,0024	00,0018	00,0019
NC	**Treino**	00,0140	00,0042	00,0132	00,0038	00,0075	00,0020	00,0073	00,0017
	Teste	00,0100	00,0072	00,0025	00,0485	00,0018	00,0306	00,0012	00,0004
SVM	**Treino**	03,4418	01,2891	04,2831	01,5325	00,6497	00,2077	00,6804	00,1860
	Teste	00,0017	00,0068	00,0187	00,0356	00,0335	00,0002	00,0026	00,0002
SGD	**Treino**	00,6506	00,0999	00,6185	00,1022	00,1078	00,0827	00,1033	00,0226
	Teste	00,0023	00,0069	00,0068	00,0021	00,0071	00,0002	00,0010	00,0002
Perceptron	**Treino**	00,6403	00,0943	00,5590	00,0874	00,0953	00,0175	00,1471	00,0222
	Teste	00,0090	00,0070	00,0017	00,0236	00,0008	00,0003	00,0248	00,0002
MLP	**Treino**	51,7928	25,1560	30,3219	43,1366	18,0916	08,8951	12,1234	13,4095
	Teste	00,0095	00,0053	00,0063	00,0056	00,0064	00,0014	00,0033	00,0022

Na Tabela 5.1 observa-se que o classificador que apresentou melhor resultado foi o KNN usando descritor GIST e redimensionamento de 128x128 (padronizado). Na Tabela 5.3, o KNN é o segundo classificador mais rápido quanto ao tempo de treinamento (em relação ao *dataset* completo com mesmo descritor e escala), porém possui o pior tempo de teste. O classificador NC possui o melhor tempo de treinamento e o terceiro melhor de teste com pouco menos de 1% de acurácia que o KNN, o que mostra que outros fatores além da acurácia do resultado devem ser considerados ao escolher um classificador. Apesar de se tratar de segundos, uma diferença de um tempo ≈700 vezes maior pode significar demais quanto se tratam de milhões de arquivos maliciosos.

5.1.2 *Dataset* Local

Na segunda etapa dos experimentos são executados os mesmos algoritmos de classificação no *dataset* local. Avaliou-se o *dataset* completo, com um grande número de famílias e uma grande variação no número de amostras por família, que é um cenário mais próximo da realidade, mas que pode interferir nos resultados por existirem grupos com poucas amostras e não possuindo, assim, tantas características fortes que descrevam a família. Para evitar isso, o *dataset* foi reduzido a um subconjunto apenas contendo as famílias que possuíam 50 exemplares ou mais. Observou-se que a taxa de acerto aumentou significativamente ao reduzir o *dataset*. Então foi executado os experimentos em um subconjunto ainda menor, com apenas dez famílias e normalizadas, reduzindo a 100 amostras por família, apresentando resultados mais representativos. Por último foi utilizado um subconjunto de 8 famílias, como usado em Makandar e Patrot (2016) e Makandar e Patrot (2017a), mostrando que a seleção de famílias e amostras para classificação pode enviesar os resultados.

É importante relembrar que esse *dataset* possui uma grande variedade de exemplares de cada família e, como visto na Figura 2.3 da Seção 2.3, variantes de famílias diferentes podem apresentar semelhança estrutural que dificulta o processo de classificação das amostras.

A Tabela 5.4 apresenta as taxas de acerto dos classificadores usados no *dataset* local completo. Foram usado os mesmos algoritmos, com os mesmos parâmetros, já usados no Malimg Dataset. Os tempos para os demais classificadores são exibidos no final da subseção na Tabela 5.8.

Tabela 5.4: Porcentagem da acurácia de classificação no *dataset* local de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128; célula cinza com a maior acurácia e vermelha com a menor).

Classificador	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[1]	34,36	14,46	18,87	04,10	27,90	11,08	10,05	29,85
LBP[1]	13,13	08,00	10,46	01,13	16,31	06,46	00,00	16,00
GIST[2]	35,08	19,49	23,18	03,69	28,51	13,33	04,10	29,54
LBP[2]	13,64	11,59	15,49	01,44	16,61	01,03	00,51	21,44

Através da tabela nota-se que todos os algoritmos atingem uma acurácia bastante baixa nesse cenário mais próximo do mundo real. Como já apontado anteriormente, isso pode acontecer devido a pouca quantidade de amostras em algumas famílias, ficando difícil caracterizá-la e por possuir muitos grupos para classificação. Porém no mundo real o número de famílias de *malware* conhecidas é muito maior que esse, portanto o esperado seria que uma boa abordagem não sofresse com esse problema.

Estabelecendo um mínimo de 50 amostras, o *dataset* foi reduzido para 37 famílias, ainda um pouco a mais que o *dataset* público, porém evitando o problema de possuir poucas amostras

para caracterizar a família. A Tabela 5.5 mostra os resultados da acurácia de classificação dos algoritmos utilizados e a Tabela 5.8 exibe os tempos de treinamento e teste dos mesmos.

Tabela 5.5: Porcentagem da acurácia de classificação no subconjunto do *dataset* local com pelo menos 50 amostras por família de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128).

Classificador	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[1]	49,13	26,09	38,84	10,72	32,32	12,03	17,39	38,98
LBP[1]	23,77	15,51	23,77	07,10	20,43	13,19	03,48	22,46
GIST[2]	48,84	29,13	43,04	15,36	35,36	26,09	20,87	41,01
LBP[2]	30,43	22,90	31,16	12,46	21,88	10,29	08,26	26,67

Nesses experimentos a classificação continua insatisfatória, porém chega a quase 50% no melhor dos casos usando KNN. Usando CNN a acurácia atinge 45,52% com média de 3021,6270 segundos de treinamento.Escolhendo as famílias mais representativas (as 10 maiores, com 100 ou mais amostras por família), foram realizados novos experimentos. Como um *dataset* não balanceado pode resultar em uma classificação errada (enviesada) (Rossow et al., 2012), optou-se por utilizar 100 amostras por família. Os resultados dos classificadores são exibidos na Tabela 5.6 e os tempos de treino e teste são exibidos na Tabela 5.9. Com o classificador CNN obteve-se uma acurácia de 76,50%. O tempo de treinamento do CNN, como já pode ser notado, é bastante superior aos demais. Para esse experimento atingiu 930,6748 segundos e 3,0792 segundos no tempo de teste.

Tabela 5.6: Porcentagem da acurácia de classificação no subconjunto do *dataset* local com as 10 famílias mais representativas de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128; célula cinza com a maior acurácia e vermelha com a menor).

Classificador	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[1]	78,00	59,00	70,00	54,00	69,00	62,00	58,00	72,00
LBP[1]	63,00	37,00	46,00	43,00	43,00	11,00	13,00	41,00
GIST[2]	78,00	73,00	70,00	63,00	61,00	57,00	52,00	66,00
LBP[2]	67,00	52,00	64,00	32,00	34,00	12,00	10,00	43,00

Usando esse novo subconjunto de dados obteve-se resultados próximos a 80% de acurácia, mostrando que essa seleção de uma quantidade menor de famílias e o balanceamento de amostras conseguem melhorar bastante os resultados de classificação. Para finalizar, utiliza-se um subconjunto menor ainda, com oito famílias (como em Makandar e Patrot (2017a) e Makandar e Patrot (2016)), e é feito novamente o balanceamento de dados para então serem aplicados os algoritmos de classificação. A Tabela 5.7 apresenta a acurácia de classificação de cada algoritmo. O classificador CNN atingiu 84,75% de acurácia com um tempo de treinamento de 654,4633 segundos e de teste 0,3134 segundos. Os tempos de treinamento e classificação dos demais classificadores são exibidos na Tabela 5.9.

Para esse subconjunto foram selecionadas oito famílias dentre as dez do experimento anterior, usando as mesmas amostras para caracterizar família, ficando claro que essa redução de famílias aumenta a taxa de acerto de classificação, o que levanta uma dúvida quanto aos experimentos realizados sob esse caráter.

Os melhores resultados de classificação em todos os experimentos foram obtidos com o classificador KNN, mostrando-se uma boa solução para o problema baseado em texturas. Em geral a escala que apresentou melhores resultados foi a padronizada, de 128x128, o que era

Tabela 5.7: Porcentagem da acurácia de classificação no subconjunto do *dataset* local com as 8 famílias mais representativas de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128; célula cinza com a maior acurácia e vermelha com a menor).

Classificador	**KNN**	**DT**	**RF**	**NC**	**SVM**	**SGD**	**Perceptron**	**MLP**
GIST[1]	87,50	66,25	81,25	67,50	66,25	68,75	57,50	77,50
LBP[1]	62,50	42,50	50,00	37,50	53,75	23,75	27,50	47,50
GIST[2]	88,75	75,00	86,25	77,50	68,75	70,00	51,25	81,25
LBP[2]	76,25	67,50	77,50	47,50	46,25	35,00	66,25	52,50

esperado, pois reduzir uma textura numa escala menor pode fazer a textura perder características relevantes, além disso apresentar a imagem de tamanho maior como entrada para o classificador reduziu o tempo de treinamento e teste, como pode ser visto nas Tabelas 5.8 e 5.9 que exibem os tempos médios de treinamento e teste dos experimentos realizados no *dataset* local.

Tabela 5.8: Tempo médio em segundos de treinamento e teste por classificador no *dataset* local ([1]: *dataset* completo, [2]: subconjunto com 50 amostras ou mais; célula cinza com o melhor tempo e vermelha com o pior).

Classificador	Escala Descritor	Literatura GIST[1]	Literatura LBP[1]	Padronizada GIST[1]	Padronizada LBP[1]	Literatura GIST[2]	Literatura LBP[2]	Padronizada GIST[2]	Padronizada LBP[2]
KNN	**Treino**	00000,5700	00000,0359	00000,5625	00000,0355	00000,0880	00000,0061	00000,0844	00000,0068
	Teste	00006,4607	00000,1939	00005,2918	00000,1966	00001,3629	00000,0432	00001,1708	00000,0292
DT	**Treino**	02205,9527	00020,1175	01939,0210	00044,4431	00004,6109	00000,1363	00004,4190	00000,1967
	Teste	00000,1537	00000,0336	00000,2183	00000,0317	00000,0008	00000,0006	00000,0009	00000,0006
RF	**Treino**	00237,0247	00012,5690	00200,4212	00022,3440	00001,5488	00000,2264	00001,3687	00000,3122
	Teste	00000,2543	00000,2129	00000,2136	00000,2391	00000,0058	00000,0055	00000,0050	00000,0061
NC	**Treino**	00000,6883	00000,7104	00000,6493	00000,7031	00000,0103	00000,0038	00000,0105	00000,0050
	Teste	00000,4334	00000,2277	00000,3335	00000,3302	00000,0026	00000,0082	00000,0019	00000,0012
SVM	**Treino**	02940,3218	01020,1608	02992,6473	00941,9732	00017,4419	00001,2744	00016,7308	00001,4659
	Teste	00000,2402	00000,1640	00000,3435	00000,1866	00000,0014	00000,0007	00000,0023	00000,0070
SGD	**Treino**	00479,4740	00115,3438	00521,3157	00119,5836	00000,6215	00000,0927	00000,6867	00000,0991
	Teste	00000,7561	00000,1445	00000,4745	00000,2572	00000,0015	00000,0068	00000,0015	00000,0067
Perceptron	**Treino**	00539,5473	00102,6126	00450,6975	00104,3392	00000,6771	00000,0909	00000,6191	00000,0920
	Teste	00003,4276	00000,1338	00000,4506	00000,1306	00000,0081	00000,0068	00000,0014	00000,0008
MLP	**Treino**	07037,6489	10953,7607	12146,2408	15815,1118	00056,9942	00042,2026	00033,1436	00019,3309
	Teste	00002,7790	00003,4318	00003,0856	00002,1197	00000,0055	00000,0047	00000,0053	00000,0025

Tabela 5.9: Tempo médio em segundos de treinamento e teste por classificador no *dataset* local ([1]: subconjunto de 10 famílias, [2]: subconjunto de 8 famílias; célula cinza com o melhor tempo e vermelha com o pior).

Classificador	Escala / Descritor	Literatura GIST[1]	Literatura LBP[1]	Padronizada GIST[1]	Padronizada LBP[1]	Literatura GIST[2]	Literatura LBP[2]	Padronizada GIST[2]	Padronizada LBP[2]
KNN	**Treino**	0,0049	0,0009	0,0051	0,0009	0,0037	0,0010	0,0059	0,0007
	Teste	0,0321	0,0025	0,0325	0,0018	0,0200	0,0025	0,0204	0,0020
DT	**Treino**	0,2327	0,0113	0,2182	0,0147	0,1455	0,0073	0,1618	0,0130
	Teste	0,0003	0,0001	0,0002	0,0002	0,0002	0,0001	0,0003	0,0222
RF	**Treino**	0,1009	0,0323	0,0937	0,0415	0,0901	0,0319	0,0687	0,0345
	Teste	0,0012	0,0010	0,0012	0,0011	0,0010	0,0010	0,0010	0,0010
NC	**Treino**	0,0014	0,0005	0,0016	0,0006	0,0013	0,0005	0,0011	0,0005
	Teste	0,0294	0,0002	0,0011	0,0002	0,0004	0,0002	0,0004	0,0333
SVM	**Treino**	0,3406	0,0442	0,3710	0,0438	0,1904	0,0342	0,2161	0,0277
	Teste	0,0328	0,0001	0,0065	0,0002	0,0003	0,0002	0,0003	0,0002
SGD	**Treino**	0,0251	0,0047	0,0229	0,0051	0,0158	0,0037	0,0143	0,0038
	Teste	0,0005	0,0001	0,0005	0,0001	0,0002	0,0002	0,0001	0,0003
Perceptron	**Treino**	0,0240	0,0046	0,0242	0,0048	0,0135	0,0030	0,0191	0,0043
	Teste	0,0005	0,0001	0,0005	0,0001	0,0002	0,0001	0,0003	0,0001
MLP	**Treino**	7,7948	4,1067	4,1832	4,1075	6,6045	1,1310	3,3722	1,6849
	Teste	0,0009	0,0005	0,0009	0,0005	0,0007	0,0004	0,0007	0,0004

Quanto ao tempo de treinamento e teste é perceptível que o algoritmo NC costuma ter o menor tempo de treinamento e um bom tempo de teste e se o tempo fosse a única variável a se analisar, provavelmente seria a melhor opção. O melhor tempo de teste normalmente é atingido pelo DT, porém é um classificador que possui um tempo de treinamento um pouco alto. Algumas técnicas como SGD e Perceptron costumam ter um tempo de classificação alto e uma baixa acurácia, não sendo considerados boas opções para classificação de *malware* por análise de textura. Quanto a taxa de acerto de classificação o melhor resultado é do algoritmo KNN que, apesar de não ter os melhores tempos, ainda assim é considerado baixo.

5.2 CLASSIFICAÇÃO DE *MALWARE* OFUSCADO

Para evitarem a detecção de AVs, são usadas diversas técnicas de ofuscação normalmente não reconhecidas através da análise estática. De acordo com Al-Anezi (2014), mais de 80% dos arquivos maliciosos utilizam alguma técnica de embalagem. A análise de textura se diz resiliente a algumas técnicas de ofuscação comuns. Para confirmar essa afirmação, foram usados compactadores comuns nos binários originais, que foram então convertidos em textura e classificados sob os mesmos critérios que os demais exemplares dos exemplos anteriores. Nesta seção serão vistos compactadores bastante simples (`TAR.GZ`, `ZIP` e `UPX`), normalmente são utilizadas técnicas mais complexas para ofuscação de *malware*. Os experimentos foram realizados apenas no subconjunto local com 8 famílias, que apresentou bons resultados de classificação e onde havia os arquivos binários originais. Os resultados de classificação são vistos em suas subseções correspondentes.

5.2.1 ZIP e TAR.GZ

O `ZIP` e o `TAR.GZ` são compressores bastante conhecidos e similares, por isso foram agrupados na mesma subseção. Utilizam algoritmos pra fazer essa compressão que procuram reduzir o tamanho do arquivo convertendo grupos de caracteres em um único caracter sempre que for possível. Quando realizadas comparações no tamanho do arquivo antes e depois da compressão, os dois reduzem aproximadamente a mesma quantidade.

De acordo com Bradley et al. (1998), o ZIP possui um algoritmo de compactação baseado no LZSS, que funciona observando a entrada de uma sequência de bytes e substitui ela por um ponteiro para a ocorrência anterior. Al-Anezi (2014) afirma que o ZIP também usa DEFLATE (combinação de LZ77 e Huffman), mas como compacta arquivos individualmente, não pode tirar vantagem de redundância de arquivos, que acontece com o TAR.GZ. O TAR combina arquivos descomprimidos e separados por cabeçalhos em um único arquivo de forma padronizada (Krintz e Calder, 2001). Usado em conjunto com o GZIP forma o TAR.GZ. Para Krintz e Calder (2001), o GZIP usa um algoritmo simples bit a bit para compactar os arquivos e, conforme Al-Anezi (2014), é baseado no algoritmo DEFLATE.

Por serem bastante simples e a técnica de compressão utilizada por ambos ser considerada resiliente à ofuscação era esperado que as taxas de acerto de classificação fossem bastante altas, similares aos resultados obtidos com o *dataset* original. As taxas de acerto usando o compressor ZIP são exibidas na Tabela 5.10, enquanto com o compressor TAR.GZ são exibidas na Tabela 5.11.

Tabela 5.10: Porcentagem da acurácia de classificação no *dataset* local comprimido com ZIP de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128; célula cinza com a maior acurácia e vermelha com a menor).

Classificador	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[1]	65,00	42,50	52,50	40,00	53,75	31,25	23,75	51,25
LBP[1]	28,75	23,75	32,25	30,00	18,75	16,25	12,50	30,00
GIST[2]	60,00	42,50	57,50	45,00	43,75	30,00	31,25	56,25
LBP[2]	30,00	35,00	35,00	30,00	26,25	13,75	12,50	20,00

Tabela 5.11: Porcentagem da acurácia de classificação no *dataset* local comprimido com TAR.GZ de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128; célula cinza com a maior acurácia e vermelha com a menor).

Classificador	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[1]	66,25	40,00	53,75	40,00	50,00	25,00	30,00	58,75
LBP[1]	27,50	23,75	31,25	31,25	22,50	12,50	12,50	37,50
GIST[2]	66,25	48,75	56,25	55,00	48,75	16,25	31,25	53,75
LBP[2]	31,25	36,25	40,00	30,00	25,00	12,50	12,50	21,25

Utilizando CNN foi atingido 45,75% de acurácia para o *dataset* utilizando ZIP e 42,13% usando TAR.GZ. Enquanto com o *dataset* original atingia-se quase 90% de acurácia, utilizando técnicas bastante simples de compressão essa taxa reduz para menos de 70%, mostrando que essa técnica não é tão resiliente à técnicas comuns de ofuscação.

5.2.2 UPX

O UPX (*Ultimate Packer for eXecutables*) é um compressor de arquivos executáveis bastante conhecido. É específico para executáveis e, uma vez comprimido, não é necessário descompactar o arquivo para executá-lo, o que evita corrompê-lo. De acordo com Al-Anezi (2014), oferece taxa de compressão mais alta que o GZIP e velocidade de descompressão mais rápida, porém os autores afirmam que por ser amplamente conhecido, é facilmente descompactado. É adicionado um pequeno módulo escrito em assembler no início do arquivo, que extrai o conteúdo do arquivo para a memória e passa o controle para ele. Por esta razão, os arquivos que usam UPX

são bastante similares. Alguns arquivos ficaram muito pequenos e ficaram fora do experimento, porém as famílias continuaram bastante balanceadas, tendo de 96 a 100 amostras por família. Na Tabela 5.12 é exibida a acurácia de classificação no *dataset*.

Tabela 5.12: Porcentagem da acurácia de classificação no *dataset* local comprimido com UPX de acordo com o descritor e escala ([1]: referente ao valor do redimensionamento usado na literatura; [2]: usando escala de 128x128).

Classificador	**KNN**	**DT**	**RF**	**NC**	**SVM**	**SGD**	**Perceptron**	**MLP**
GIST[1]	12,99	12,99	12,99	12,99	12,99	12,99	12,99	12,99
LBP[1]	12,99	12,99	12,99	12,99	12,99	12,99	12,99	12,99
GIST[2]	12,99	12,99	12,99	12,99	12,99	12,99	12,99	12,99
LBP[2]	12,99	12,99	12,99	12,99	12,99	12,99	12,99	11,69

Pela tabela é possível observar que a acurácia de classificação ficou bastante similar para todos os classificadores, mas observando as matrizes de confusão no Apêndice A é mais fácil de observar o motivo. Todas as amostras são classificadas na mesma família. Essa porcentagem de acerto representa as amostras que pertencem de fato a essa família, porém não há distinção pelo classificador. Com CNN atingiu 12,50% e a matriz de confusão mostra que não agrupa em apenas uma família, mas também não apresenta uma boa classificação.

A Figura 5.2 mostra texturas geradas para diferentes famílias ofuscadas com UPX. Por ser um compressor de arquivos executáveis, ele insere informações do compressor para que haja a descompressão automaticamente ao ser executado, ou seja, o arquivo deixa de ter suas características pois torna-se um arquivo UPX. Também é importante lembrar que quando comprimido, o arquivo reduz significativamente seu tamanho e por esse motivo é tão difícil encontrar similaridades entre as famílias.

(a) Família Agobot

(b) Família Bancos

(c) Família Rbot

(d) Família Sdbot

Figura 5.2: Textura de variantes de diferentes famílias do *dataset* local comprimidas com UPX.

Quando são utilizadas técnicas de ofuscação em binários de *malware*, os atacantes costumam utilizar técnicas mais robustas do que a usada no UPX. Se a classificação usando análise de texturas não consegue distinguir os arquivos que usam essas técnicas simples, não é possível dizer que é resiliente a técnicas de ofuscação.

5.3 CONCLUSÃO

Neste capítulo foram apresentados os resultados de classificação de diferentes algoritmos de classificação aplicados em dois *datasets* e em subconjuntos dos mesmos para mostrar que os bons resultados apresentados na literatura podem ter sido enviesados através de uma seleção de amostras para classificação.

No *dataset* público foi mostrado que realmente é possível atingir altas taxas de classificação com os algoritmos utiliados na literatura. Foi observado que o uso do descritor global apresentou melhores resultados, assim como o redimensionamento aqui proposto. Quando usado o subconjunto do *dataset* conforme proposto por alguns autores, a classificação chegou a apresentar 100% de acurácia, mostrando como a seleção de dados pode inteferir no resultado final.

Ao utilizar um novo *dataset*, as diferentes técnicas de classificação apresentam um resultado bastante ruim. Esses resultados começam a melhorar conforme os dados para classificação são selecionados. Em todos os conjuntos de dados o classificador KNN apresenta melhor taxa de acerto, quanto ao descritor, obteve-se melhor resultado através do GIST e a escala de 128x128 apresentou a melhor acurácia na maioria dos casos.

Para comprovar a eficácia de classificação de arquivos binários ofuscados foram utilizados alguns compressores simples e os experimentos apontam que a classificação usando análise de texturas não é resiliente às técnicas de ofuscação.

6 CONCLUSÕES E TRABALHOS FUTUROS

A classificação de *malware* em famílias é um desafio na área de segurança computacional há anos. Quando realizada de forma acurada consegue deter as atividades maliciosas mais rápida e efetivamente, inclusive para exemplares até então desconhecidos. Ao longo dos anos foram propostas diversas maneiras de realizar essa classificação, normalmente baseada em atributos extraídos da análise dinâmica ou análise estática. Nesta dissertação, avaliou-se técnicas baseadas em texturas, uma forma relativamente nova de analisar os binários que se mostrou promissora.

Vários fatores podem influenciar nos resultados de classificação com essa abordagem. Por esse motivo foram realizados experimentos variando diversos parâmetros, por exemplo, a escala da textura. Esta, redimensionada inicialmente, apresentou melhores resultados quando usado um tamanho maior (128x128); avaliando-se o tempo de treinamento e teste é possível notar que usar a textura em tamanho maior não resultou em um maior tempo de classificação, sendo que em alguns casos, o tempo inclusive diminuiu. Outras variações de parâmetros para a avaliação são citadas abaixo.

Foram usados dois descritores de textura, um global e um local, pois apesar da literatura utilizar em geral o descritor global, o descritor local poderia evitar uma das desvantagens apresentadas por Nataraj et al. (2013) referente a troca de ordem de seções do binário. Os descritores foram usados como apresentados na literatura e o descritor global, GIST, teve uma taxa de acerto superior. Isso pode ter ocorrido pelo tamanho do vetor de características gerado, bem maior usando GIST, deixando as características da textura mais evidentes. Uma mudança nos parâmetros do descritor local para gerar um vetor de características maior, poderia melhorar esses resultados.

Quanto aos classificadores, foram usados diferentes algoritmos de classificação, desde os mais simples e tradicionais (como KNN e DT), até alguns mais recentes e um pouco mais complexos (como RF e CNN). Porém, o algoritmo que apresentou maior taxa de acerto em todos os experimentos foi o KNN, cuja operação é bastante simples. Foram usados os mesmos parâmetros propostos na literatura e uma mudança nos parâmetros dos classificadores poderia resultar numa acurácia mais elevada.

Os experimentos realizados no *dataset* público exibem os resultados replicados da literatura, mostrando que os algoritmos realmente funcionam como os autores apresentaram em suas publicações, porém quando utilizado um *dataset* com famílias não tão grandes e conhecidas, esse resultado cai expressivamente, sugerindo uma seleção dos dados por aqueles que são melhor diferenciáveis. Isso pode ser observado quando é feita uma seleção de amostras do *dataset* local, a qual incorre em aumento significativo na acurácia de classificação devido ao número limitado de famílias e de as famílias serem bem diferentes entre si. O uso de um *dataset* selecionado produz resultados de classificação bastante atraentes, porém enviesados. Os experimentos mostram que, em um cenário real, as técnicas de classificação de famílias de *malware* baseadas na análise de texturas não apresentam bons resultados, não sendo recomendada sua utilização.

Portanto, respondendo a questão inicial do problema, “em um cenário real, com milhares de exemplares e famílias de *malware*, utilizar unicamente a estratégia de análise de texturas para classificação de *malware* pode resolver esse problema de classificação com as vantagens

apontadas na literatura?", os experimentos apontam que essa abordagem não pode resolver o problema de classificação de *malware* em famílias em um cenário real, pois a própria extração das texturas ser feita sobre a conversão de um binário tem suas limitações. Por exemplo, não é possível observar o comportamento de execução desses binários.

Uma das vantagens apontada pelos autores refere-se à resiliência quanto ao uso de técnicas de ofuscação por programas maliciosos. Nesta dissertação, avaliou-se essa vantagem em potencial utilizando-se apenas compressores comuns nas amostras (`UPX, ZIP, TAR.GZ`). Num cenário real, é esperado que sejam utilizados algoritmos muito mais sofisticados para ofuscar os binários, próprios para programas maliciosos. Entretanto, quando usadas essas técnicas simples, a taxa de acerto dos classificadores já diminui bastante. Usando UPX, praticamente nenhum classificador conseguiu diferenciar as amostras em famílias, atribuindo o mesmo rótulo a todas elas. Com isso, mostrou-se que a classificação por texturas, ao contrário do que é clamado em trabalhos relacionados, não pode ser considerada resiliente às técnicas de ofuscação comumente utilizadas em *malware*.

Para melhorar as técnicas utilizadas, como já afirmado, é possível alterar alguns parâmetros dos classificadores e descritores, o que traria um certo ganho na taxa de acerto de classificação. Técnicas de engenharia reversa poderiam ser aplicadas nos binários ofuscados antes de convertê-los em textura, porém aumentariam o custo computacional (memória e tempo de processamento).

Apesar de a análise de texturas ser rápida e conseguir classificar corretamente algumas famílias de *malware*, não é possível utilizá-la sozinha para classificação, uma vez que o método não apresenta características suficientes para descrever as famílias. Pesquisas futuras podem incluir o uso da análise de textura para detecção e classificação de *malware* para identificar variantes bastante conhecidas em conjunto com algum outro tipo de análise para atribuir um maior grau de certeza, talvez considerando a análise de textura apenas quando a textura for extremamente similar.

É possível também modificar a forma de representar o binário. A utilização da visualização em 2D pode perder informação por definir uma largura resultando em alguns casos numa descontinuidade da sequência. Identificar inicialmente o tamanho das instruções ou fazer previamente uma filtragem de *bytes* ou instruções repetidas pode auxiliar. A representação em três dimensões pode trazer uma informação mais rica, ou, até mesmo uma representação em uma dimensão pode resultar numa melhor visualização, visto que evitaria o problema da descontinuidade e poderia ser usado com o algoritmo de classificação shapelet que possui resultados interessantes na literatura. Arquivos muito pequenos não são convertidos em textura, portanto essa abordagem em uma dimensão também solucionaria esse problema.

Referências

Agarap, A. F. e Pepito, F. J. H. (2017). Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (svm) for malware classification. *arXiv preprint arXiv:1801.00318*.

Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M. e Giacinto, G. (2016). Novel feature extraction, selection and fusion for effective malware family classification. Em *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, páginas 183–194, New York, NY, USA. ACM.

Al-Anezi, M. M. K. (2014). Generic packing detection using several complexity analysis for accurate malware detection. *International Journal Advanced Computer Science*, 5(1).

Anderson, B., Storlie, C. e Lane, T. (2012). Improving malware classification: bridging the static/dynamic gap. Em *Proceedings of the 5th ACM workshop on Security and artificial intelligence*, páginas 3–14. ACM.

Avila, G. V. (2017). Análise de sentimento para textos curtos. Dissertação de Mestrado, Fundação Getulio Vargas, Escola de Matemática Aplicada, Rio de Janeiro.

Awad, R. A. e Sayre, K. D. (2016). Automatic clustering of malware variants. Em *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*, páginas 298–303. IEEE.

Bertolini, D., Oliveira, L. S., Justino, E. e Sabourin, R. (2013). Texture-based descriptors for writer identification and verification. *Expert Systems with Applications*, 40(6):2069–2080.

Bradley, Q., Horspool, R. N. e Vitek, J. (1998). Jazz: An efficient compressed format for java archive files. Em *Proceedings of the 1998 Conference of the Centre for Advanced Studies on Collaborative Research*, CASCON '98, páginas 7–. IBM Press.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Camilo, A. E., Grégio, A. e Santos, R. D. (2016). Identifying compromised systems through correlation of suspicious traffic from malware behavioral analysis. Em *SPIE Defense+ Security*, páginas 982606–982606. International Society for Optics and Photonics.

Carlin, D., O'Kane, P. e Sezer, S. (2017). Dynamic analysis of malware using run-time opcodes. Em *Data Analytics and Decision Support for Cybersecurity*, páginas 99–125. Springer.

Cepeda, C., Tien, D. L. C. e Ordóñez, P. (2016). Feature selection and improving classification performance for malware detection. Em *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on*, páginas 560–566. IEEE.

Conti, G., Bratus, S., Shubina, A., Lichtenberg, A., Ragsdale, R., Perez-Alemany, R., Sangster, B. e Supan, M. (2010a). A visual study of primitive binary fragment types. *White Paper, Black Hat USA*.

Conti, G., Bratus, S., Shubina, A., Sangster, B., Ragsdale, R., Supan, M., Lichtenberg, A. e Perez-Alemany, R. (2010b). Automated mapping of large binary objects using primitive fragment type classification. *digital investigation*, 7:S3–S12.

Conti, G., Dean, E., Sinda, M. e Sangster, B. (2008). Visual reverse engineering of binary and data files. *Visualization for Computer Security*, páginas 1–17.

Cortes, C. e Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Costa, Y. M., Oliveira, L., Koerich, A. L., Gouyon, F. e Martins, J. (2012). Music genre classification using lbp textural features. *Signal Processing*, 92(11):2723–2737.

Damodaran, A., Di Troia, F., Visaggio, C. A., Austin, T. H. e Stamp, M. (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12.

Egele, M., Scholte, T., Kirda, E. e Kruegel, C. (2008). A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv.*, 44(2):6:1–6:42.

Fix, E. e Hodges Jr, J. L. (1951). Discriminatory analysis-nonparametric discrimination: consistency properties. Relatório técnico, California Univ Berkeley.

Gadhiya, S. e Bhavsar, K. (2013). Techniques for malware analysis. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4).

Gandotra, E., Bansal, D. e Sofat, S. (2014). Malware analysis and classification: A survey. *Journal of Information Security*, 5(02):56.

Gardner, M. W. e Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14):2627–2636.

Grégio, A. R., Fernandes Filho, D. S., Afonso, V. M., Santos, R. D., Jino, M. e de Geus, P. L. (2011). Behavioral analysis of malicious code through network traffic and system call monitoring. Em *Evolutionary and Bio-Inspired Computation: Theory and Applications V*, volume 8059, página 80590O. International Society for Optics and Photonics.

Han, K., Lim, J. H. e Im, E. G. (2013). Malware analysis method using visualization of binary files. Em *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, páginas 317–321. ACM.

Han, K. S., Lim, J. H., Kang, B. e Im, E. G. (2015). Malware analysis using visualized images and entropy graphs. *International Journal of Information Security*, 14(1):1–14.

He, K., Zhang, X., Ren, S. e Sun, J. (2016). Deep residual learning for image recognition. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 770–778.

Ho, T. K. (1995). Random decision forests. Em *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, páginas 278–282. IEEE.

Kabanga, E. K. e Kim, C. H. (2017). Malware images classification using convolutional neural network. *Journal of Computer and Communications*, 6(01):153.

Kosmidis, K. (2017). Machine learning and images for malware detection and classification. Dissertação de Mestrado, School of Science Technology, Thessaloniki - Greece.

Kosmidis, K. e Kalloniatis, C. (2017). Machine learning and images for malware detection and classification. Em *Proceedings of the 21st Pan-Hellenic Conference on Informatics*, página 5. ACM.

Krintz, C. e Calder, B. (2001). Reducing delay with dynamic selection of compression formats. Em *hpdc*, página 0266. IEEE.

Lab., A. K. (2018). History of malicious programs. `https://securelist.com/threats/history-of-malicious-programs/`. Acessado em 27/04/2018.

Laks (2014). Sarvam blog. `http://sarvamblog.blogspot.com.br`. Acessado em 25/08/2017.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. e Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Levner, I. (2005). Feature selection and nearest centroid classification for protein mass spectrometry. *BMC bioinformatics*, 6(1):68.

Li, P., Liu, L., Gao, D. e Reiter, M. K. (2010). On challenges in evaluating malware clustering. Em Jha, S., Sommer, R. e Kreibich, C., editores, *Recent Advances in Intrusion Detection*, páginas 238–255, Berlin, Heidelberg. Springer Berlin Heidelberg.

Luo, J.-S. e Lo, D. C.-T. (2017). Binary malware image classification using machine learning with local binary pattern. Em *Big Data (Big Data), 2017 IEEE International Conference on*, páginas 4664–4667. IEEE.

Makandar, A. e Patrot, A. (2015a). Malware analysis and classification using artificial neural network. Em *2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)*, páginas 1–6.

Makandar, A. e Patrot, A. (2015b). Malware image analysis and classification using support vector machine. *International Journal of Trends in Computer Science and Engineering*, 4(5):01–03.

Makandar, A. e Patrot, A. (2016). An approach to analysis of malware using supervised learning classification. Em *International Conference on Recent Trends in Engineering, Science Technology*.

Makandar, A. e Patrot, A. (2017a). Malware class recognition using image processing techniques. Em *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*, páginas 76–80.

Makandar, A. e Patrot, A. (2017b). Wavelet statistical feature based malware class recognition and classification using supervised learning classifier. *Oriental Journal of Computer Science and Technology*, 10(2):400–406.

Makandar, A. e Patrot, A. (2018). Trojan malware image pattern classification. Em Guru, D. S., Vasudev, T., Chethan, H. e Kumar, Y. S., editores, *Proceedings of International Conference on Cognition and Recognition*, páginas 253–262, Singapore. Springer Singapore.

Manjunath, B. S., Ohm, J.-R., Vasudevan, V. V. e Yamada, A. (2001). Color and texture descriptors. *IEEE Transactions on circuits and systems for video technology*, 11(6):703–715.

Mathur, K. e Hiranwal, S. (2013). A survey on techniques in detection and analyzing malware executables. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4).

McIntyre, R. M. e Blashfield, R. K. (1980). A nearest-centroid technique for evaluating the minimum-variance clustering procedure. *Multivariate Behavioral Research*, 15(2):225–238.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Naidu, V. e Narayanan, A. (2016). Using different substitution matrices in a string-matching technique for identifying viral polymorphic malware variants. Em *Evolutionary Computation (CEC), 2016 IEEE Congress on*, páginas 2903–2910. IEEE.

Nanni, L., Lumini, A. e Brahnam, S. (2012). Survey on lbp based texture descriptors for image classification. *Expert Systems with Applications*, 39(3):3634 – 3641.

Nataraj, L. (2015). *A Signal Processing Approach To Malware Analysis*. University of California, Santa Barbara.

Nataraj, L., Karthikeyan, S., Jacob, G. e Manjunath, B. (2011a). Malware images: visualization and automatic classification. Em *Proceedings of the 8th international symposium on visualization for cyber security*, página 4. ACM.

Nataraj, L., Kirat, D., Manjunath, B. e Vigna, G. (2013). Sarvam: Search and retrieval of malware. Em *Proceedings of the Annual Computer Security Conference (ACSAC) Worshop on Next Generation Malware Attacks and Defense (NGMAD)*.

Nataraj, L., Yegneswaran, V., Porras, P. e Zhang, J. (2011b). A comparative assessment of malware classification using binary texture analysis and dynamic analysis. Em *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, páginas 21–30. ACM.

Nath, H. V. e Mehtre, B. M. (2014). Static malware analysis using machine learning methods. Em Martínez Pérez, G., Thampi, S. M., Ko, R. e Shu, L., editores, *Recent Trends in Computer Networks and Distributed Systems Security*, páginas 440–450, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ojala, T., Pietikainen, M. e Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(7):971–987.

Oliva, A. e Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. e Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Rezende, E., Ruppert, G., Carvalho, T., Ramos, F. e de Geus, P. (2017). Malicious software classification using transfer learning of resnet-50 deep neural network. Em *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, páginas 1011–1014.

Rezende, E., Ruppert, G., Carvalho, T., Theophilo, A., Ramos, F. e Geus, P. d. (2018). Malicious software classification using vgg16 deep neural network's bottleneck features. Em Latifi, S., editor, *Information Technology - New Generations*, páginas 51–59, Cham. Springer International Publishing.

Rieck, K., Holz, T., Willems, C., Düssel, P. e Laskov, P. (2008). Learning and classification of malware behavior. Em *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, páginas 108–125. Springer.

Rieck, K., Trinius, P., Willems, C. e Holz, T. (2011). Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668.

Robbins, H. e Monro, S. (1985). A stochastic approximation method. Em *Herbert Robbins Selected Papers*, páginas 102–109. Springer.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H. e Steen, M. v. (2012). Prudent practices for designing malware experiments: Status quo and outlook. Em *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, páginas 65–79, Washington, DC, USA. IEEE Computer Society.

Rumelhart, D. E., Hinton, G. E. e Williams, R. J. (1985). Learning internal representations by error propagation. Relatório técnico, California Univ San Diego La Jolla Inst for Cognitive Science.

Safavian, S. R. e Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674.

Sampaio, R. e Mancini, M. (2007). Estudos de revisão sistemática: um guia para síntese criteriosa da evidência científica. *Revista brasileira de fisioterapia*, 11(1):83–89.

Sebastián, M., Rivera, R., Kotzias, P. e Caballero, J. (2016). Avclass: A tool for massive malware labeling. Em *International Symposium on Research in Attacks, Intrusions, and Defenses*, páginas 230–253. Springer.

Shafiq, M. Z., Khayam, S. A. e Farooq, M. (2008). Embedded malware detection using markov n-grams. *Lecture Notes in Computer Science*, 5137:88–107.

Shaid, S. Z. M. e Maarof, M. A. (2014). Malware behavior image for malware variant identification. Em *Biometrics and Security Technologies (ISBAST), 2014 International Symposium on*, páginas 238–243. IEEE.

Sikorski, M. e Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, San Francisco, CA, USA, 1st edition.

Simon, P. e Uma, V. (2018). Review of texture descriptors for texture classification. Em *Data Engineering and Intelligent Computing*, páginas 159–176. Springer.

Singh, A. (2017). Malware classification using image representation. Dissertação de Mestrado, Department of Computer Science and Engineering - Indian Institute of Technology Kanpur, Kanpur, UP, India.

Symantec (2017). 2017 internet security threat report. `https://digitalhubshare.symantec.com/content/dam/Atlantis/campaigns-and-launches/FY17/Threat%20Protection/ISTR22_Main-FINAL-JUN8.pdf?aid=elq_`. Acessado em 25/07/2017.

Symantec (2018). 2018 internet security threat report. `https://resource.elq.symantec.com/LP=5840?cid=70138000000rm1eAAA`. Acessado em 02/05/2018.

Thakare, V. S., Patil, N. N. e Sonawane, J. S. (2013). Survey on image texture classification techniques. *International Journal of Advancements in Technology*, 4(1):97–104.

Uppal, D., Mehra, V. e Verma, V. (2014). Basic survey on malware analysis, tools and techniques. *International Journal on Computational Sciences & Applications (IJCSA) Vol*, 4:103–111.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T. e the scikit-image contributors (2014). scikit-image: image processing in Python. *PeerJ*, 2:e453.

VanHoudnos, N., Casey, W., French, D., Lindauer, B., Kanal, E., Wright, E., Woods, B., Moon, S. e Jansen, P. (2017). This malware looks familiar: Laymen identify malware run-time similarity with chernoff faces and stick figures.

Vemuri, V., editor (1988). *Artificial Neural Networks: Theoretical Concepts*. IEEE Computer Society Press, Los Alamitos, CA, USA.

VirusTotal (2017). Virustotal. `https://www.virustotal.com/#/home/upload`. Acessado em 25/04/2017.

Wagner, M., Fischer, F., Luh, R., Haberson, A., Rind, A., Keim, D. A., Aigner, W., Borgo, R., Ganovelli, F. e Viola, I. (2015). A survey of visualization systems for malware analysis. Em *EG Conference on Visualization (EuroVis)-STARs*, páginas 105–125.

Willems, C., Holz, T. e Freiling, F. (2007). Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy*, 5(2).

Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y. e Sakuma, J. (2018a). Malware analysis of imaged binary samples by convolutional neural network with attention mechanism. Em *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, páginas 127–134. ACM.

Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y. e Sakuma, J. (2018b). Malware analysis of imaged binary samples by convolutional neural network with attention mechanism. Em *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, CODASPY '18, páginas 127–134, New York, NY, USA. ACM.

Yue, S. (2017). Imbalanced malware images classification: a CNN based approach. *CoRR*, abs/1708.08042.

Zhang, J., Qin, Z., Yin, H., Ou, L., Xiao, S. e Hu, Y. (2016). Malware variant detection using opcode image recognition with small training sets. Em *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*, páginas 1–9. IEEE.

APÊNDICE A: RESULTADOS DE CLASSIFICAÇÃO

Os experimentos realizados no Capítulo 5 geraram matrizes de confusão que mostram o resultado de classificação um pouco mais detalhado, exibindo as amostras de quais famílias foram classificadas em outras famílias.

A.1 MALIMG *DATASET*

O primeiro classificador a ser analisado é o KNN, usado por Nataraj et al. (2011a), com redimensionamento inicial proposto de 64x64 e com k = 3. A Figura A.1 mostra a matriz de confusão usando descritor GIST, enquanto a Figura A.2 usando LBP. As Figuras A.3 e A.4 mostram a matriz de confusão usando descritor GIST e LBP, respectivamente, porém com redimensionamento de 128x128.

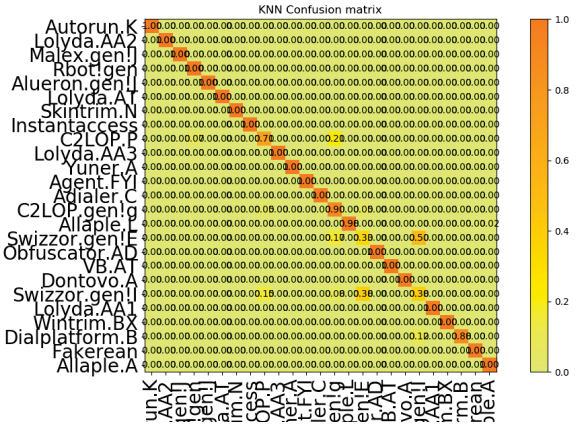


Figura A.1: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 no Malimg Dataset.

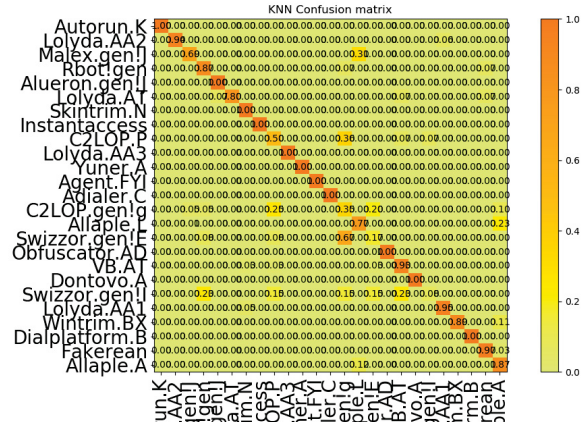


Figura A.2: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 no Malimg Dataset.

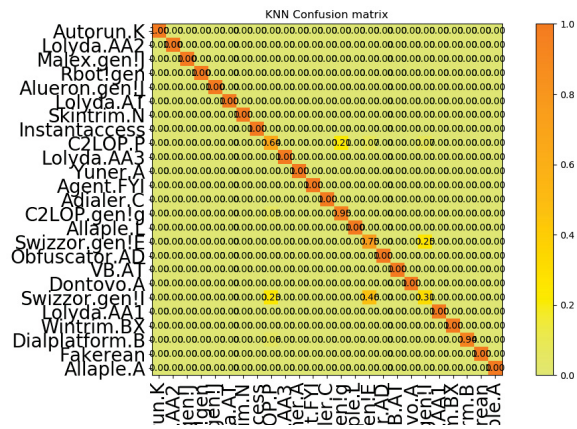


Figura A.3: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 no Malimg Dataset.

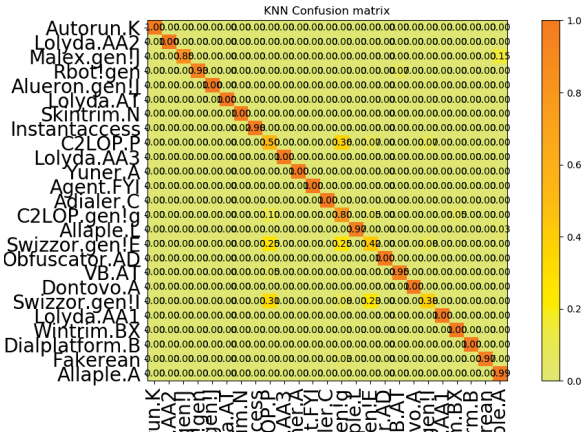


Figura A.4: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 no Malimg Dataset.

O mesmo procedimento foi realizado para o classificador *Decision Trees*, porém com redimensionamento inicial de 32x32, conforme utilizado em Kosmidis e Kalloniatis (2017). Foram utilizados os valores *default* para esse classificador. As Figuras A.5 e A.6 mostram as matrizes de redimensionamento 32x32 para descritores GIST e LBP, enquanto as Figuras A.7 e A.8 de 128x128.

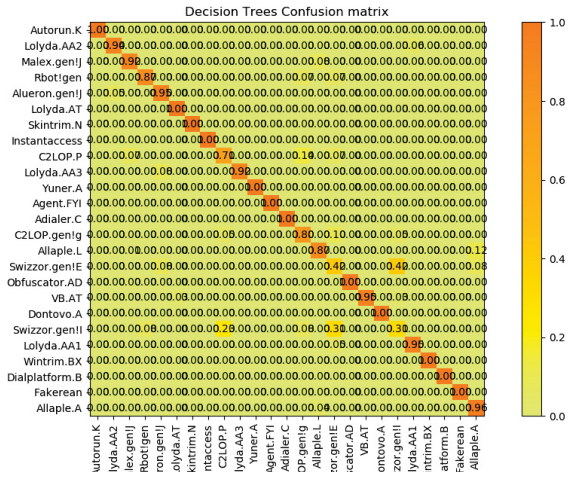


Figura A.5: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 no Malimg Dataset.

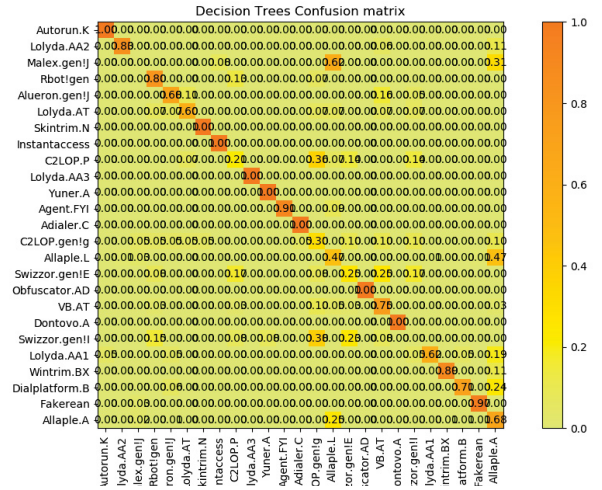


Figura A.6: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 no Malimg Dataset.

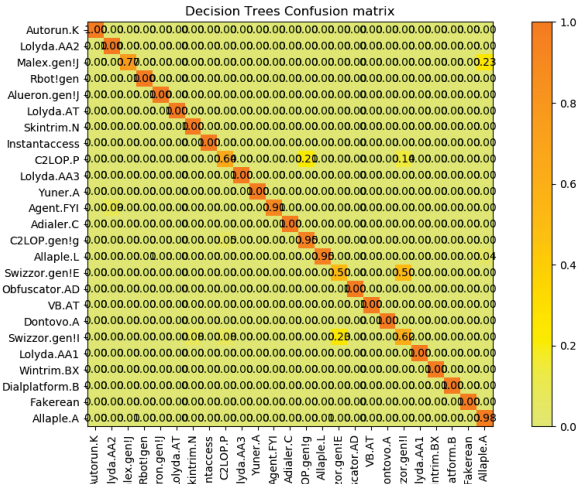


Figura A.7: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 no Malimg Dataset.

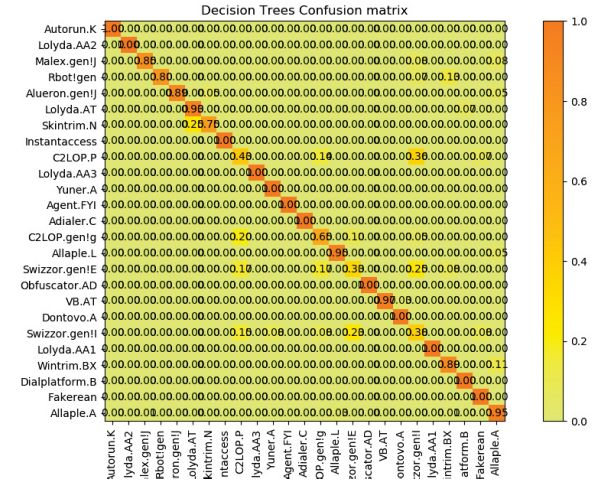


Figura A.8: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 no Malimg Dataset.

Random Forest também foi utilizado por Kosmidis e Kalloniatis (2017), portanto o redimensionamento inicial também foi de 32x32 e foram usados os valores *default* para o classificador. As Figuras A.9 e A.10 mostram a matriz de confusão com redimensionamento 32x32 e as Figuras A.11 e A.12 com redimensionamento 128x128 para os dois descritores.

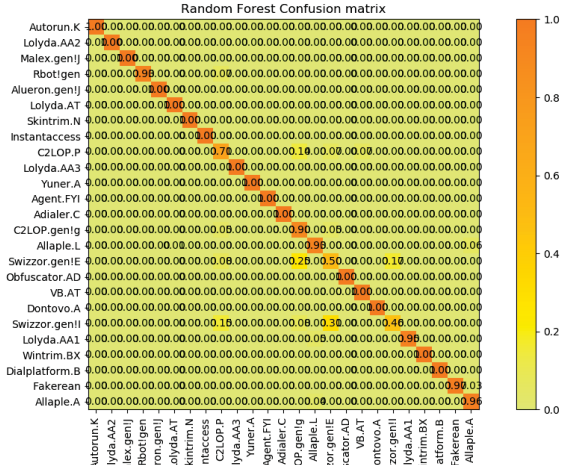


Figura A.9: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 no Malimg Dataset.

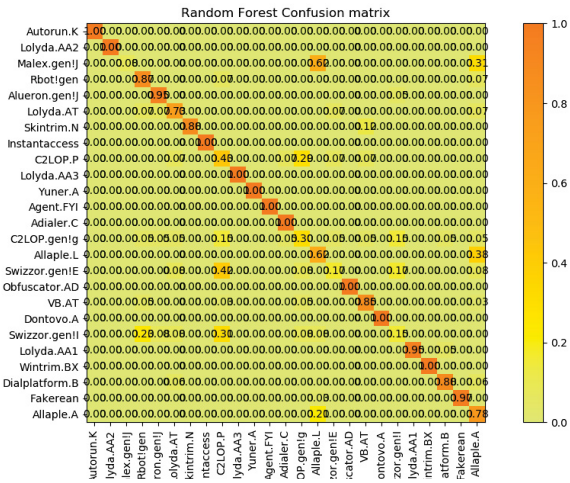


Figura A.10: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 no Malimg Dataset.

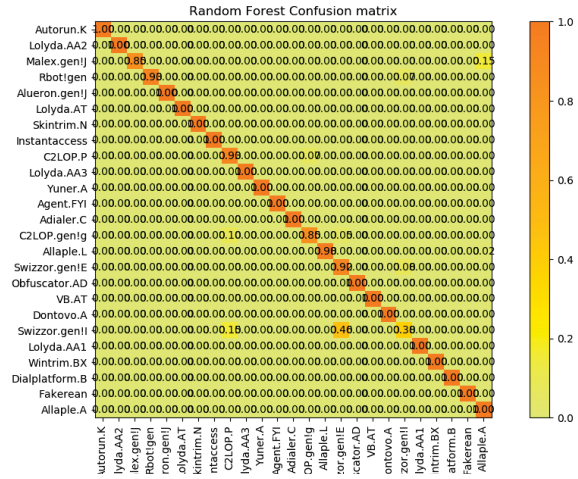


Figura A.11: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 no Malimg Dataset.

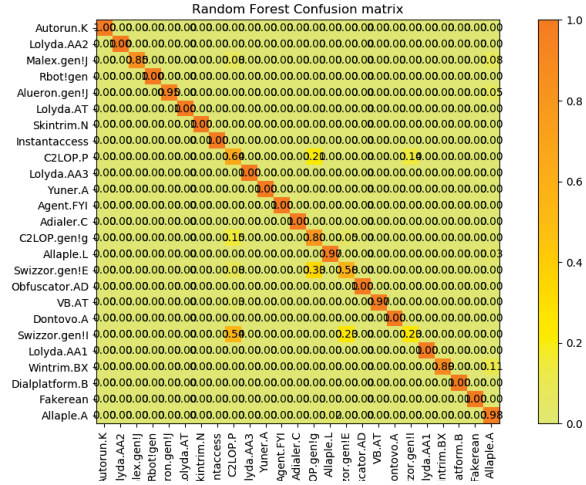


Figura A.12: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 no Malimg Dataset.

Outro classificador utilizado por Kosmidis e Kalloniatis (2017) é o *Nearest Centroid* que também foram usados os valores *default* em uma escala 32x32. As matrizes confusão geradas por esse classificador são apresentadas nas Figuras A.13, A.14, A.15 e A.16.

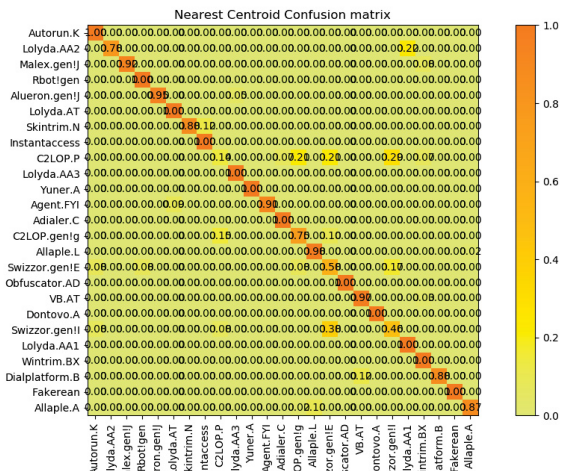


Figura A.13: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 no Malimg Dataset.

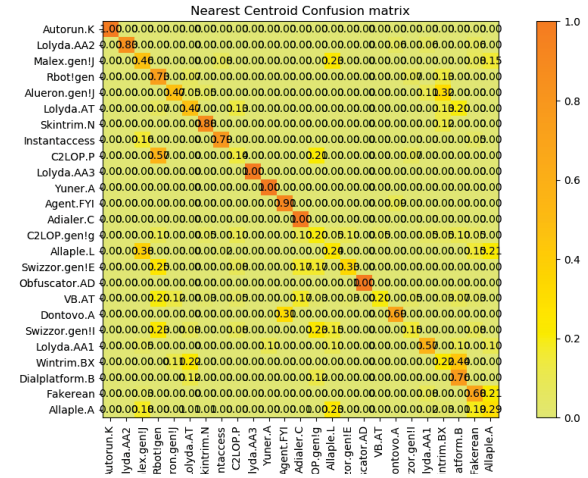


Figura A.14: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 no Malimg Dataset.

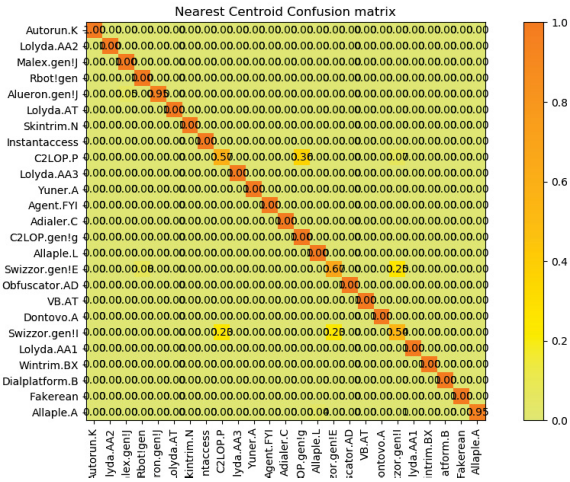


Figura A.15: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 no Malimg Dataset.

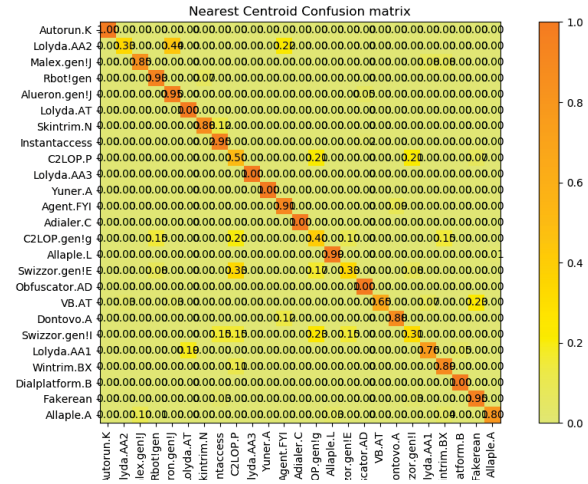


Figura A.16: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 no Malimg Dataset.

O classificador SVM foi utilizado por alguns autores na literatura. Para esses experimentos foi utilizada a classe `LinearSVC` também com os valores *default*. O melhor resultado apresentado foi por Makandar e Patrot (2017a) com um redimensionamento de 64x64, utilizado nesses experimentos. As Figuras A.17 e A.18 mostram as matrizes confusões usando o descritor GIST e o LBP sob essas condições. O redimensionamento padrão gerou as matrizes confusões das Figuras A.19 e A.20.

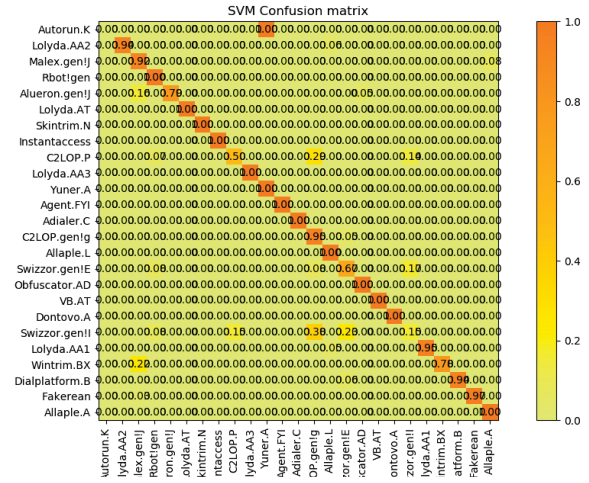


Figura A.17: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 no Malimg Dataset.

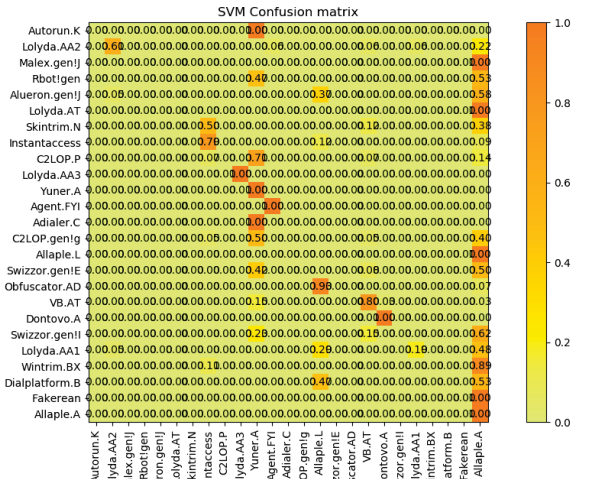


Figura A.18: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 no Malimg Dataset.

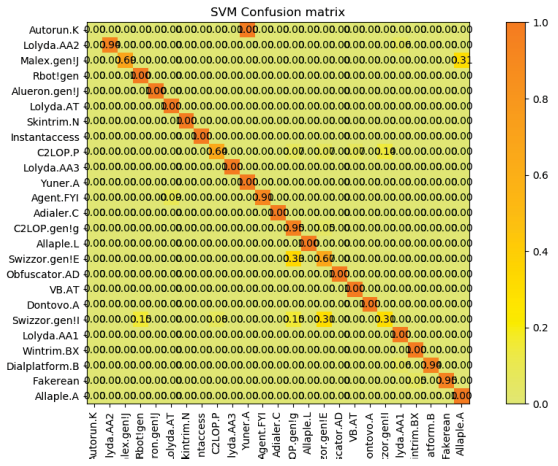


Figura A.19: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 no Malimg Dataset.

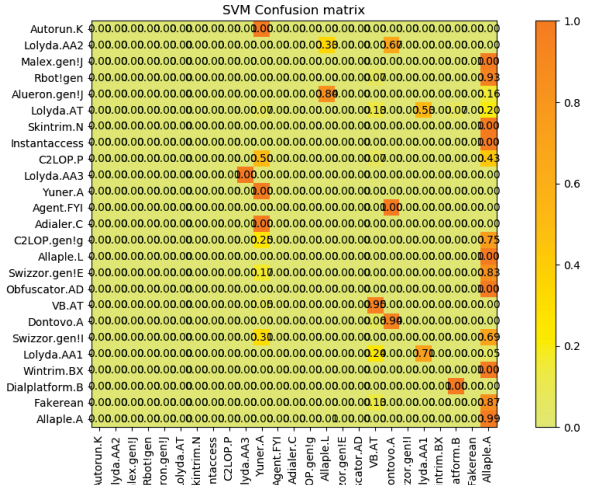


Figura A.20: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 no Malimg Dataset.

Para o SGD, usado por Kosmidis e Kalloniatis (2017), foi usado loss="hinge", penalty="l2"em uma escala 32x32, gerando as matrizes das Figuras A.21 e A.22, enquanto com a escala 128x128 gerou as das Figuras A.23 e A.24.

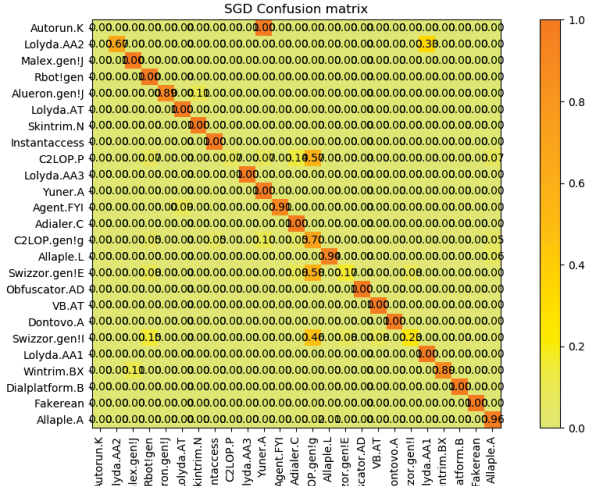


Figura A.21: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 no Malimg Dataset.

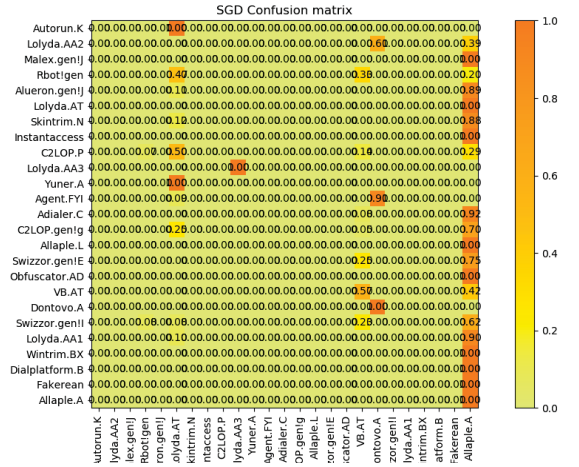


Figura A.22: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 no Malimg Dataset.

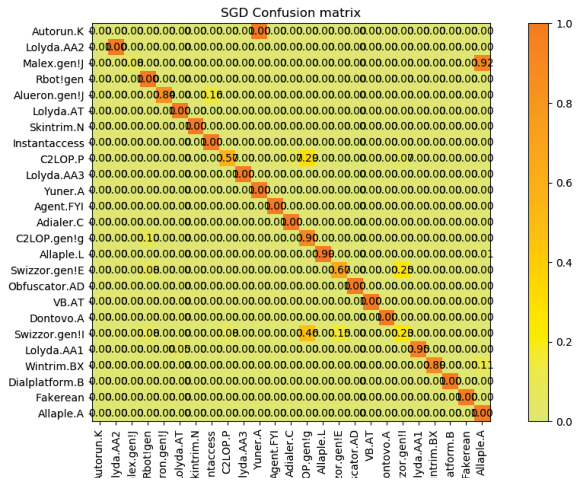


Figura A.23: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 no Malimg Dataset.

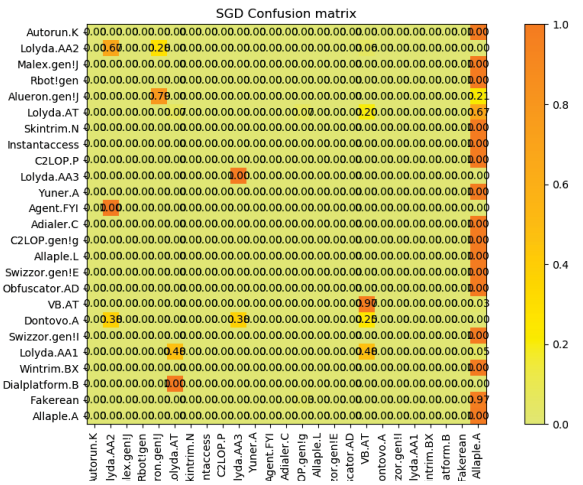


Figura A.24: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 no Malimg Dataset.

Para o Perceptron também foi usado os parâmetros atribuidos em Kosmidis e Kalloniatis (2017), gerando as matrizes de confusão das Figuras A.25, A.26, A.27 e A.28.

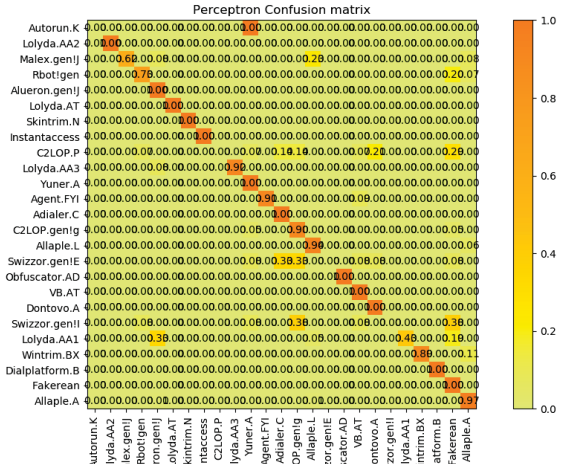


Figura A.25: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 no Malimg Dataset.

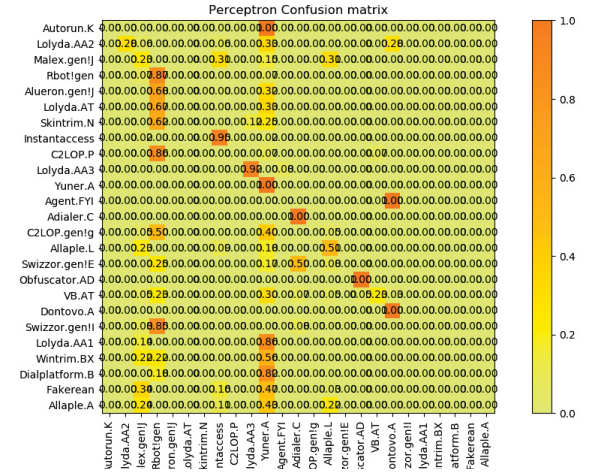


Figura A.26: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 no Malimg Dataset.

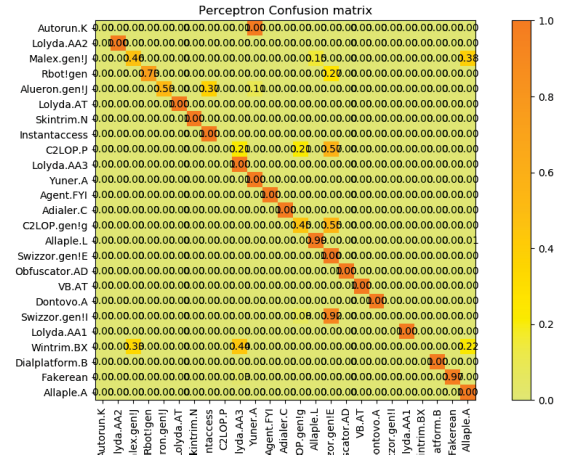


Figura A.27: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 no Malimg Dataset.

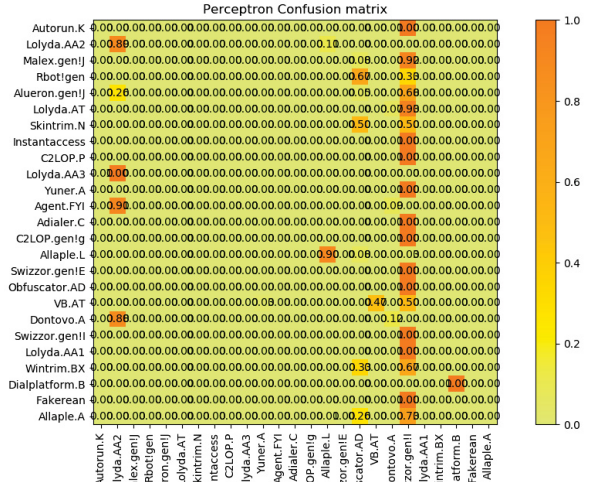


Figura A.28: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 no Malimg Dataset.

O MLP também foi aplicado com os valores *default* nas escalas 32x32 e 128x128 gerando as matrizes de confusão das Figuras A.29, A.30, A.31 e A.32.

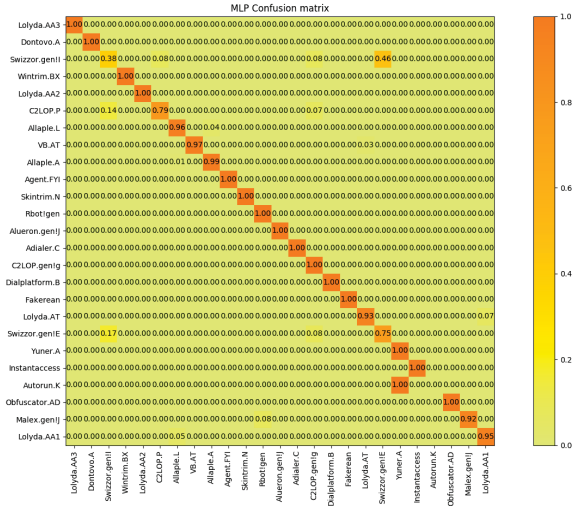


Figura A.29: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 no Malimg Dataset.

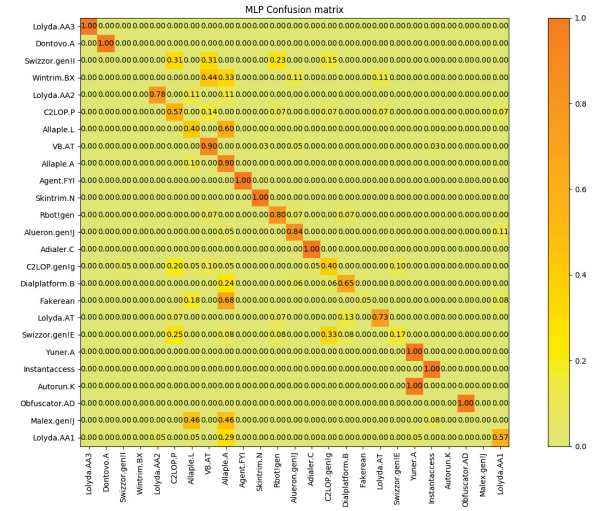


Figura A.30: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 no Malimg Dataset.

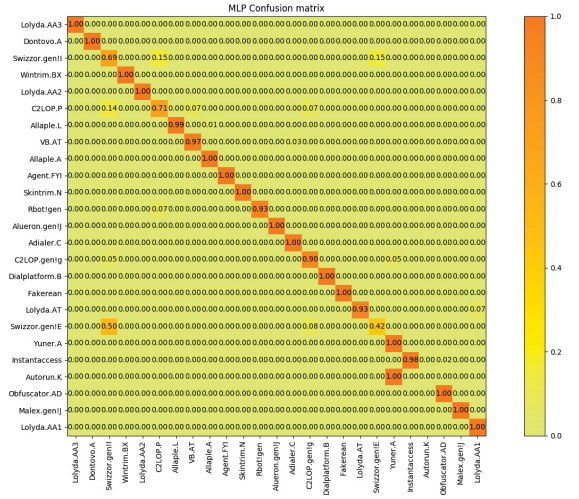


Figura A.31: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 no Malimg Dataset.

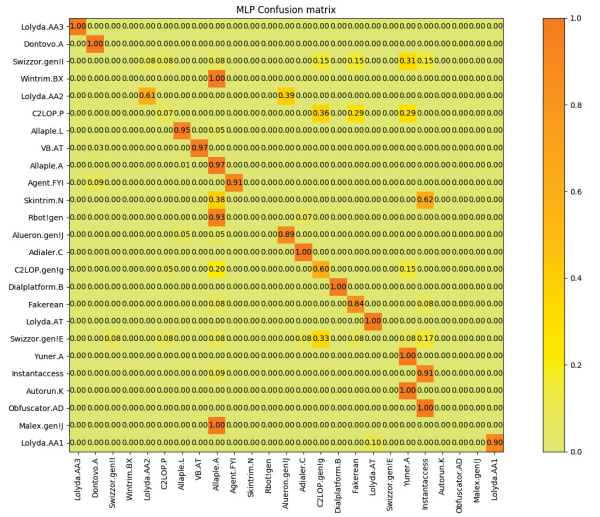


Figura A.32: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 no Malimg Dataset.

Para o CNN foi usado o mesmo algoritmo proposto por Rezende et al. (2017), devido a isso não foi possível utilizar o redimensionamento de 128x128 por ser considerado muito pequeno. Neste caso foi usado somente o proposto pela literatura, gerando a matriz de confusão da Figura A.33.

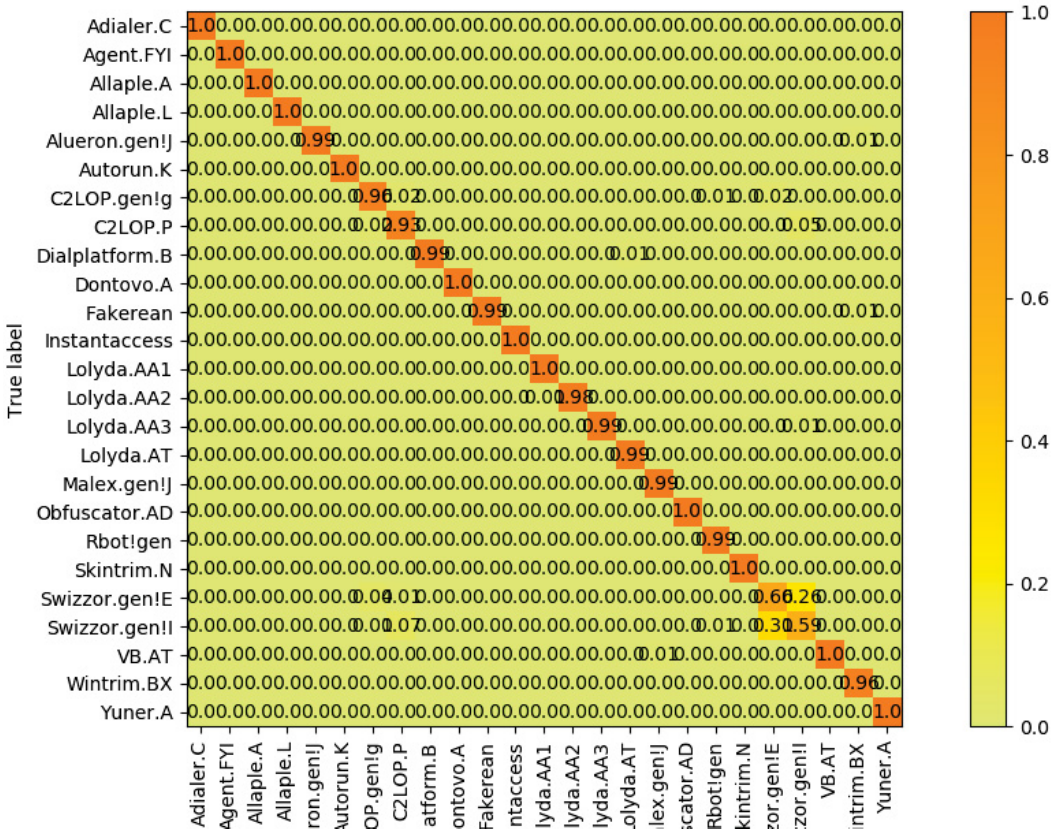


Figura A.33: Matriz de confusão do classificador CNN no Malimg Dataset.

Para este *dataset*, dentre as famílias que são classificadas incorretamente, em geral, em todos os classificadores, observa-se que as famílias `Swizzor.gen!E` e `Swizzor.gen!I` costumam ser classificadas trocadas, assim como `C2LOP.gen!g` e `C2LOP.P`, mas por serem variações das mesmas famílias, pode ser considerado um erro justo. Alguns erros entre as famílias `Swizzor` e `C2LOP` também são observados, além da família `Autorun` que era classificada de maneira errada como `Yuner` e `Malex` confundia com as variações de `Allaple`.

Na seleção do subconjunto utilizado para classificação por Makandar e Patrot (2016), essas famílias que mais eram classificadas erradas são descartadas. Em nossos experimentos, no subconjunto do *dataset* Malimg foram executados os mesmos algoritmos. As matrizes de confusão são exibidas abaixo por classificador.

- **KNN.**

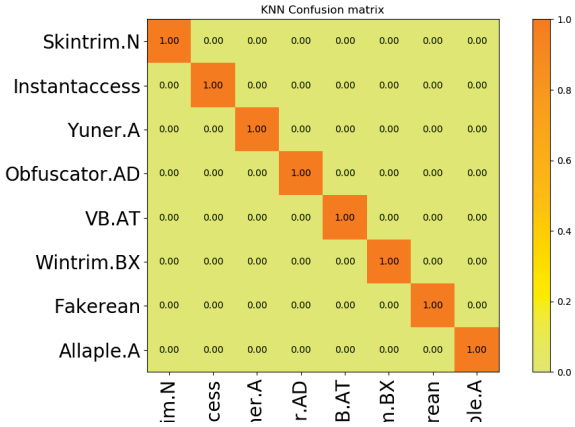


Figura A.34: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

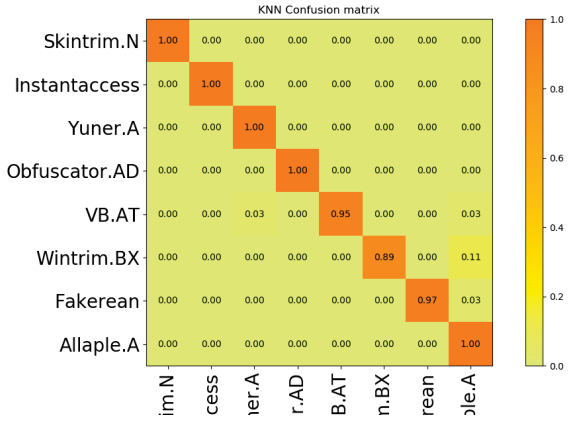


Figura A.35: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

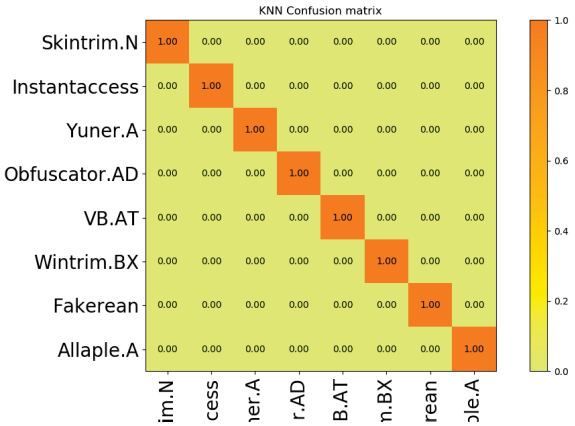


Figura A.36: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

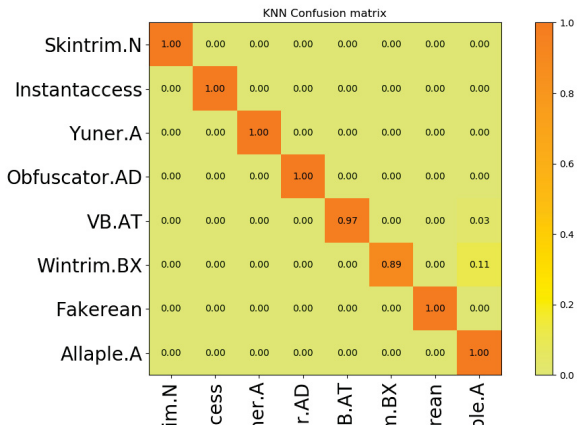


Figura A.37: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Decision Trees.**

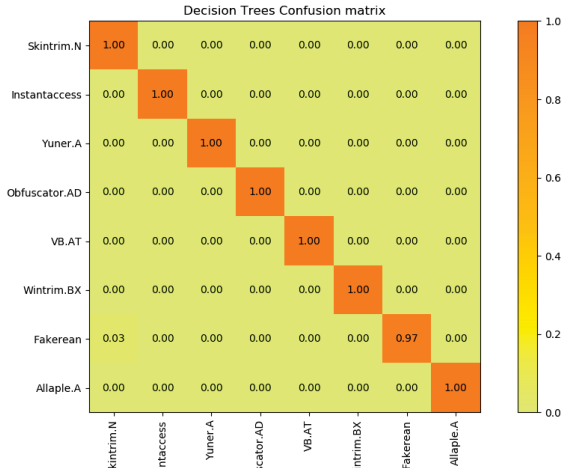


Figura A.38: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

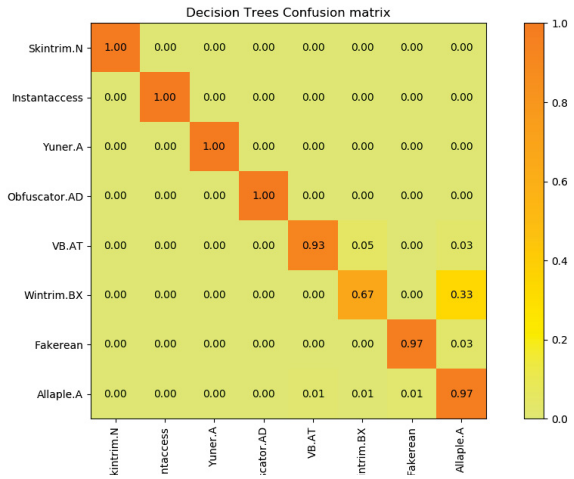


Figura A.39: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

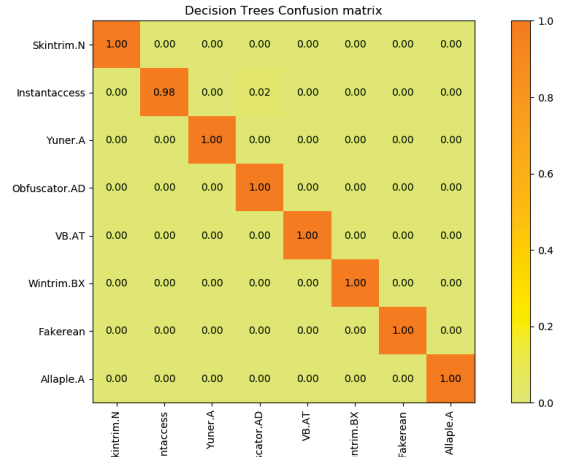


Figura A.40: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

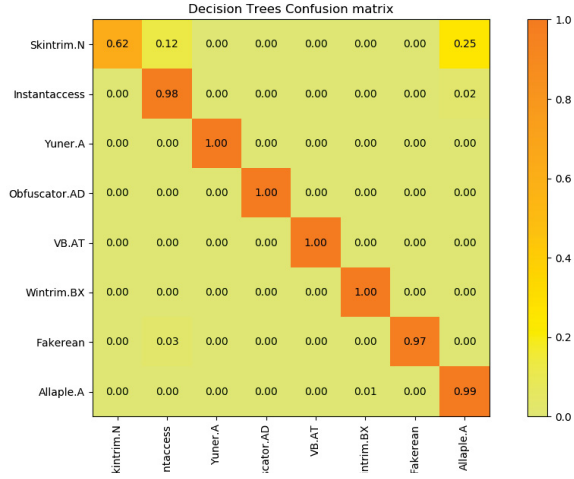


Figura A.41: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Random Forest.**

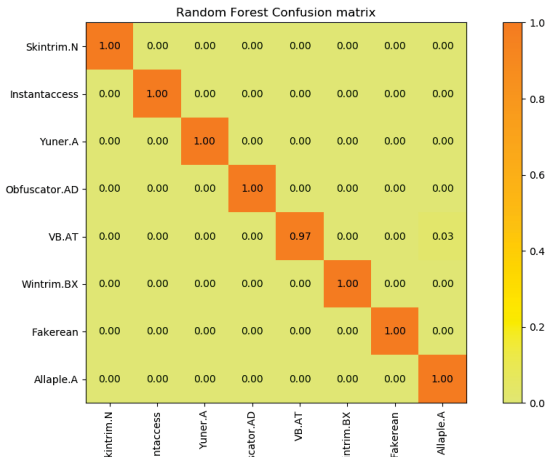


Figura A.42: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

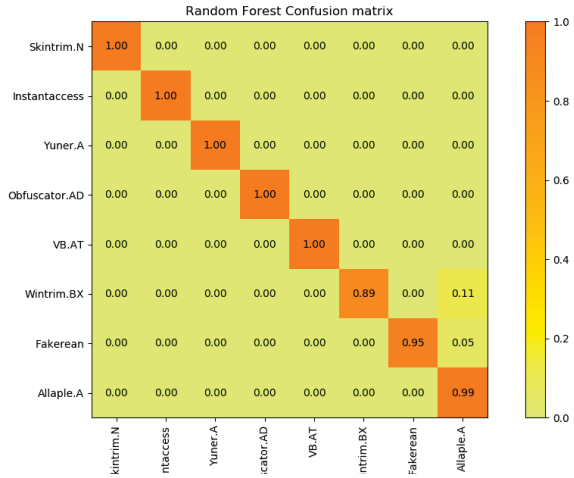


Figura A.43: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

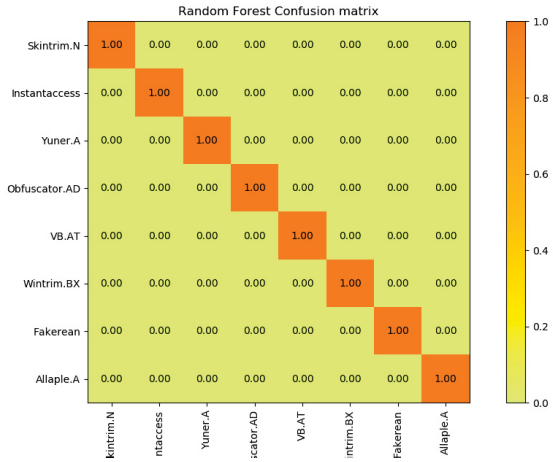


Figura A.44: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

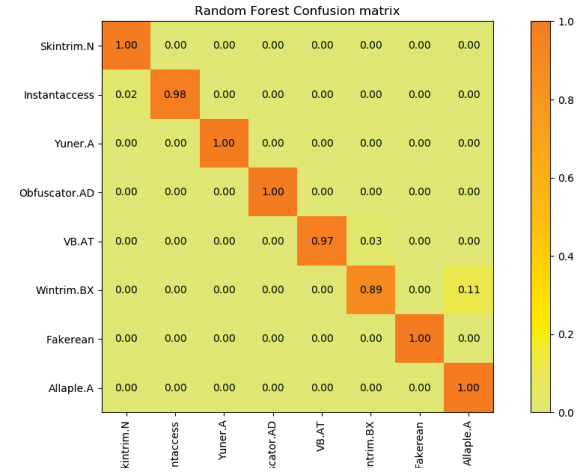


Figura A.45: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Nearest Centroid**

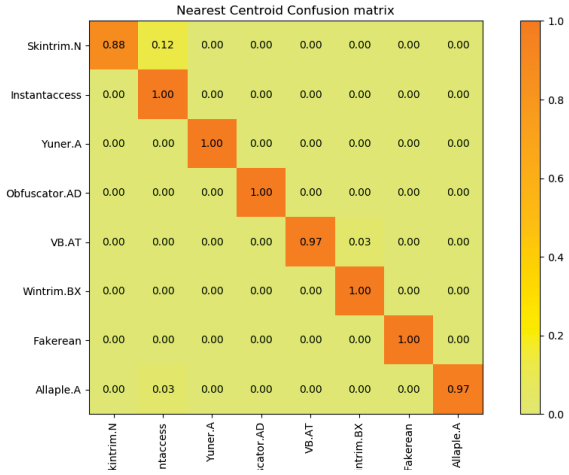


Figura A.46: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

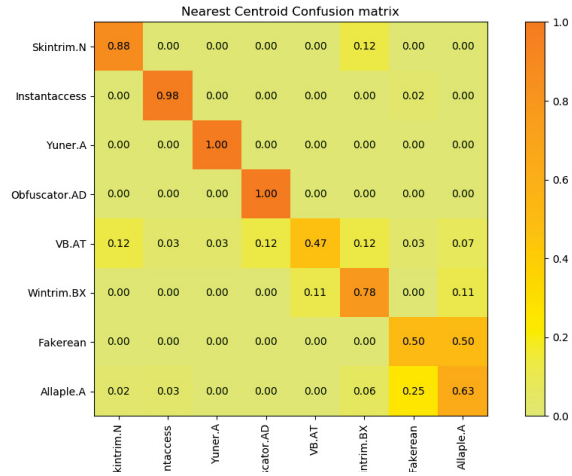


Figura A.47: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

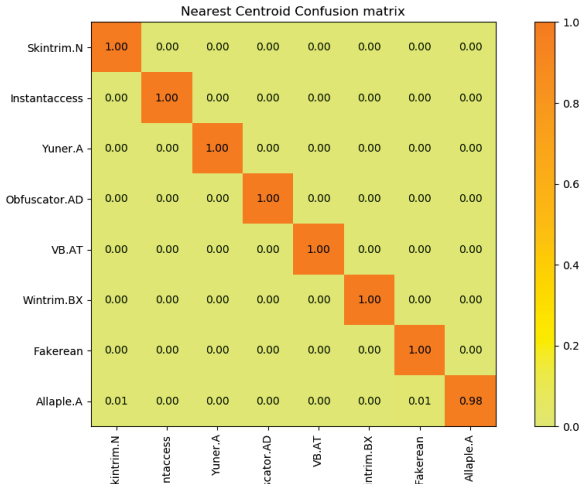


Figura A.48: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

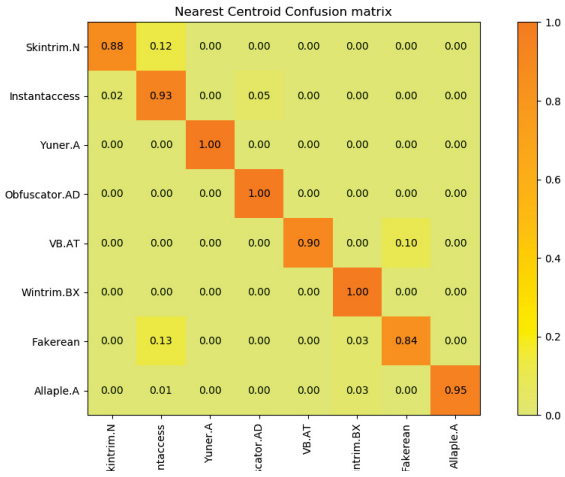


Figura A.49: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SVM.**

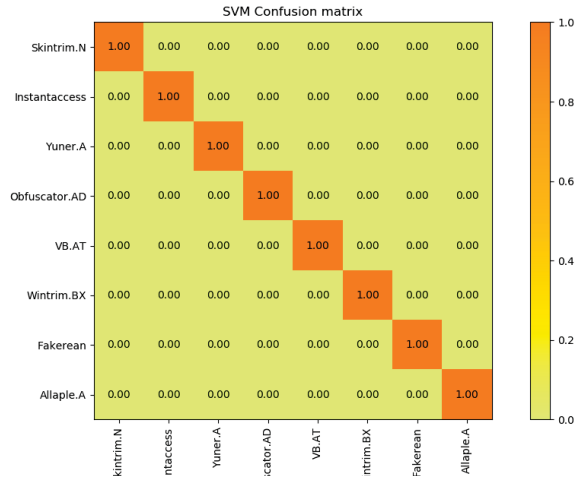


Figura A.50: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

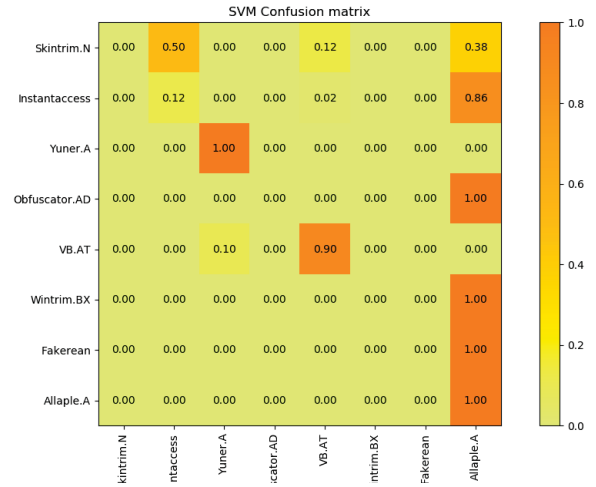


Figura A.51: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

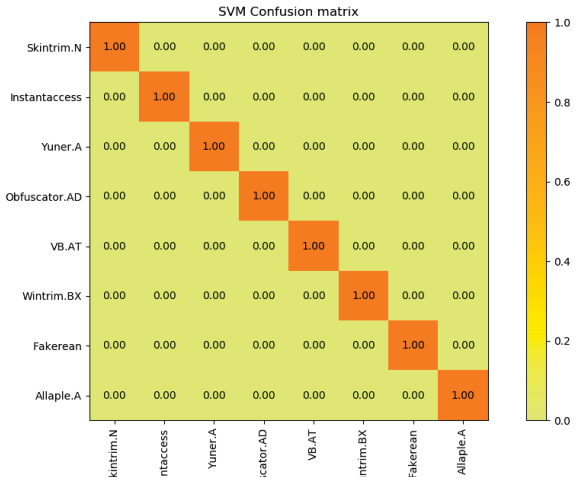


Figura A.52: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

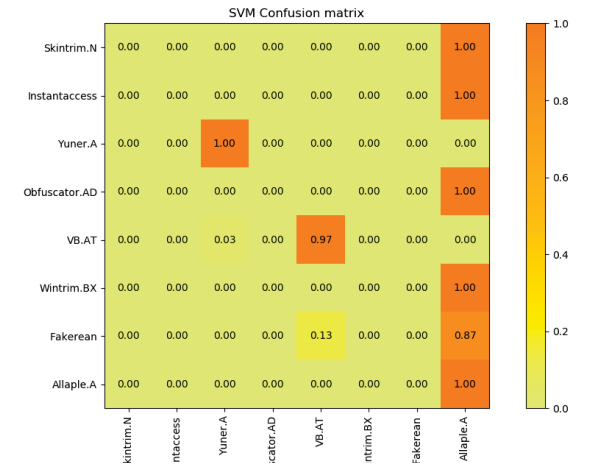


Figura A.53: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SGD.**

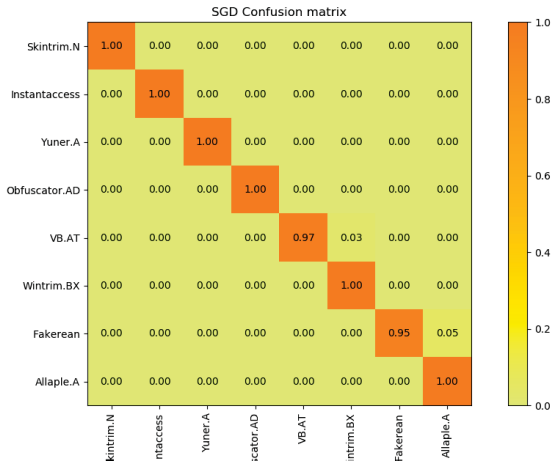


Figura A.54: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

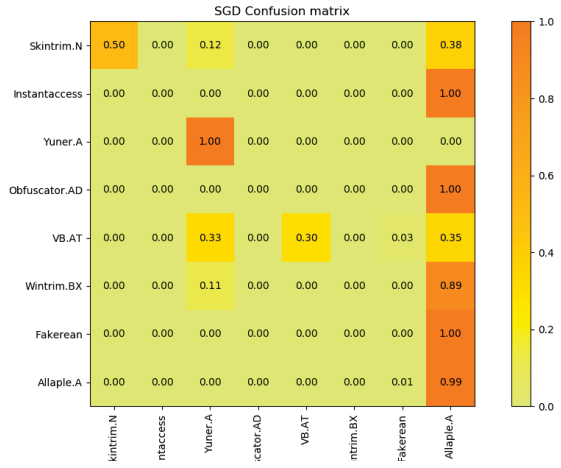


Figura A.55: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

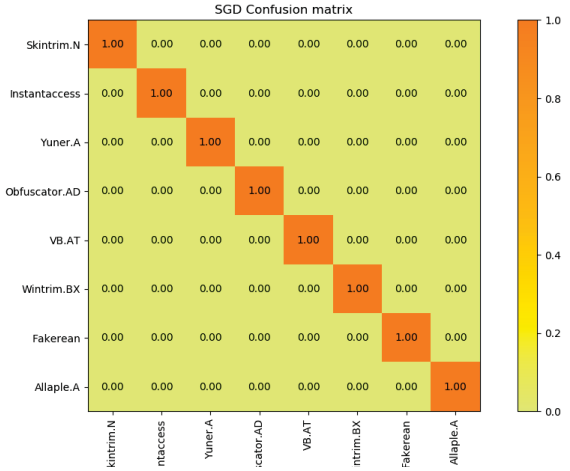


Figura A.56: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

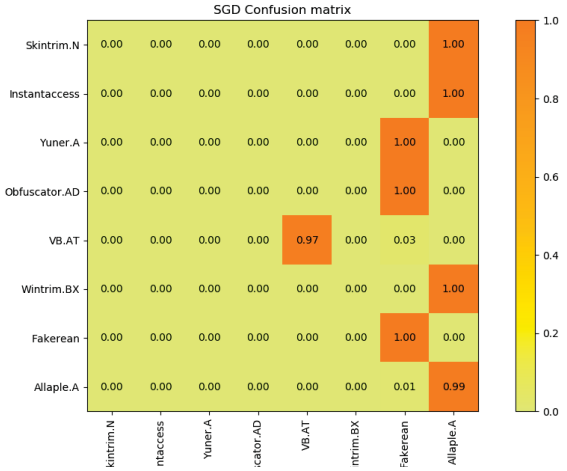


Figura A.57: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Perceptron.**

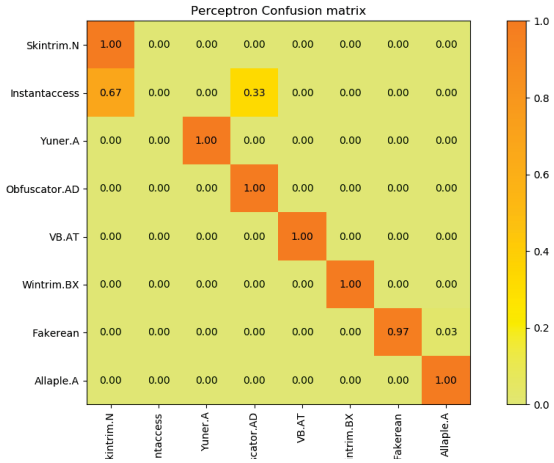


Figura A.58: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

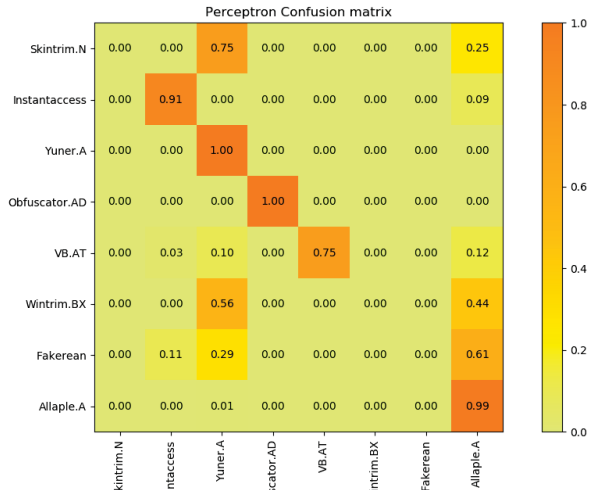


Figura A.59: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

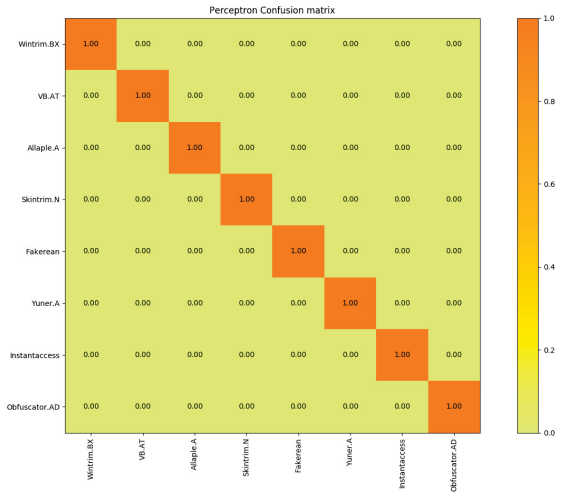


Figura A.60: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

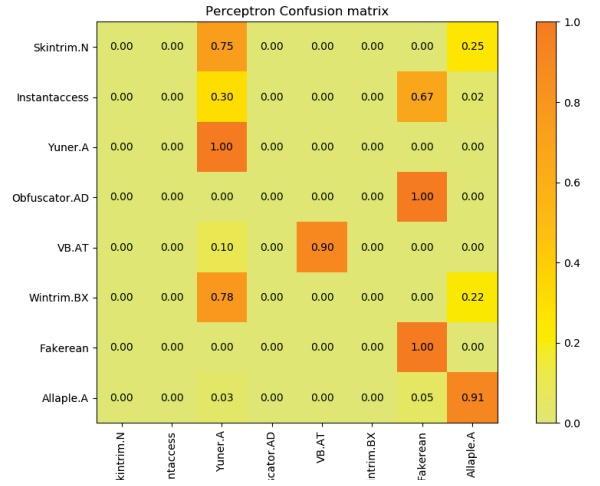


Figura A.61: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **MLP.**

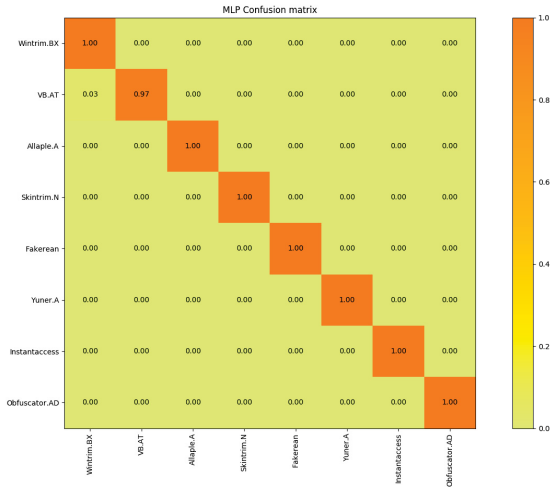


Figura A.62: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

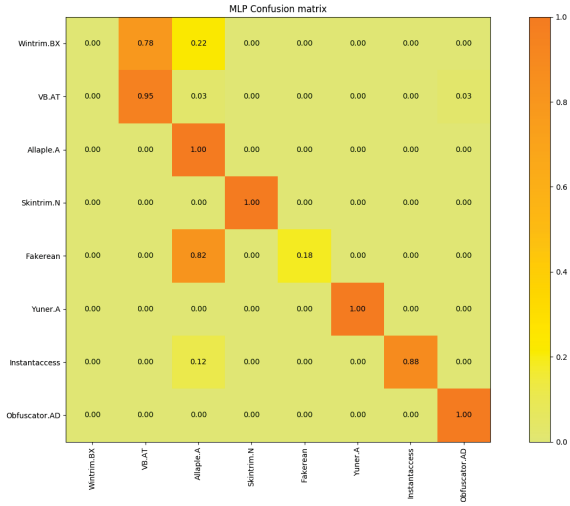


Figura A.63: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

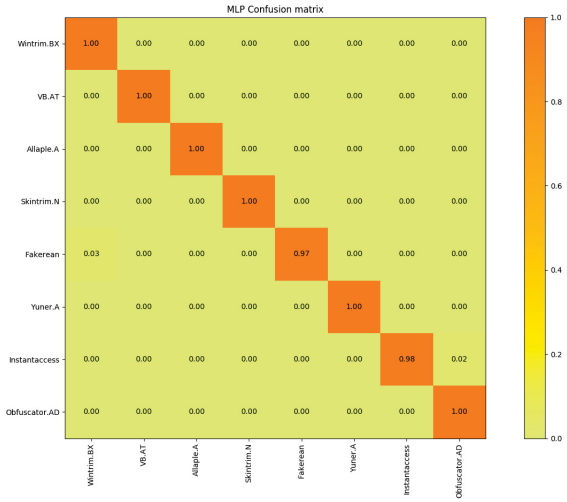


Figura A.64: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

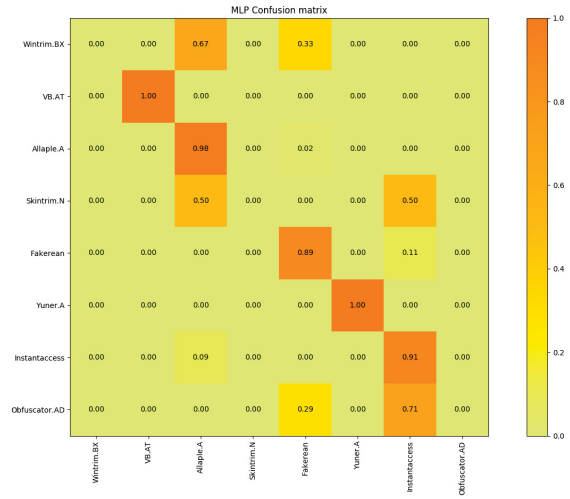


Figura A.65: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **CNN.**

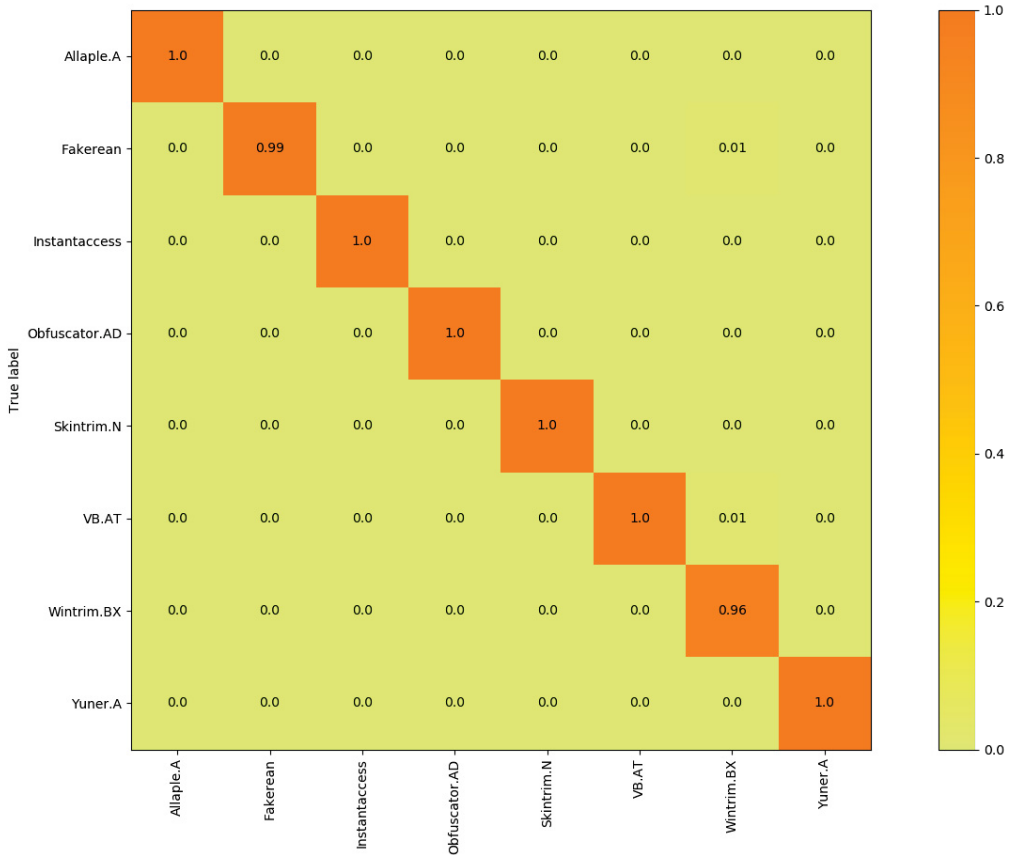


Figura A.66: Matriz de confusão do classificador CNN no subconjunto do Malimg Dataset.

Com a seleção de famílias os resultados chegam a quase 100% na maioria dos experimentos usando GIST. Porém é possível notar que o descritor local LBP não apresenta resultados tão interessantes.

A.2 *DATASET* LOCAL

Para validar as técnicas de classificação baseadas em análise de texturas foi utilizado um *dataset* local, com maior número de amostras e famílias. As matrizes de confusão do *dataset* local aplicados nos mesmos algoritmos são exibidos a seguir.

- **KNN.**

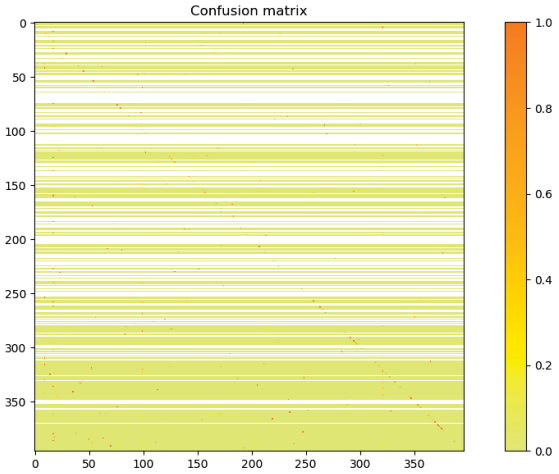


Figura A.67: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

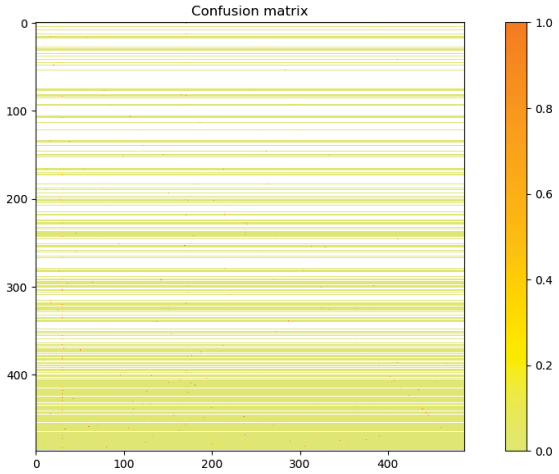


Figura A.68: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

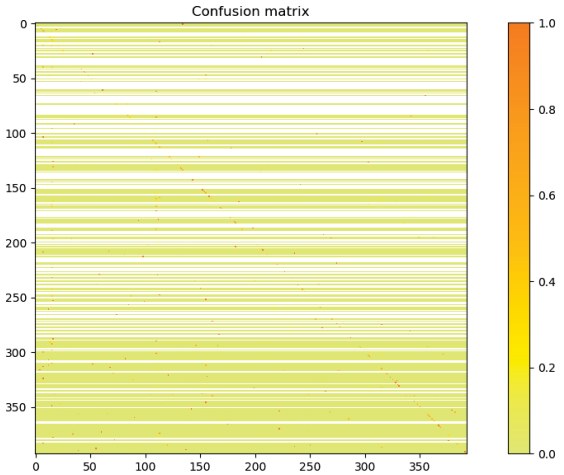


Figura A.69: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

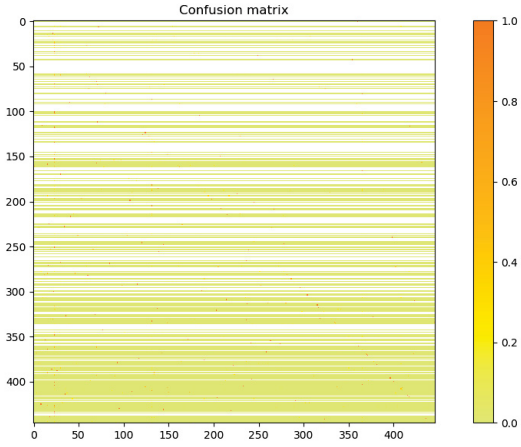


Figura A.70: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Decision Trees.**

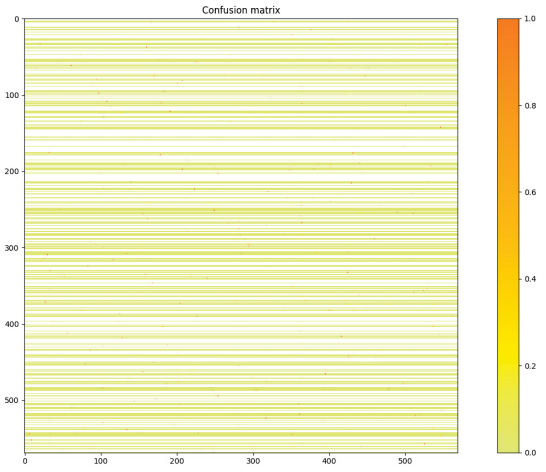


Figura A.71: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

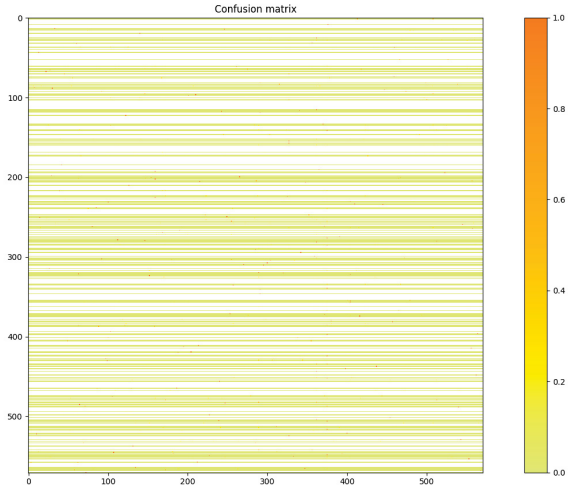


Figura A.72: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

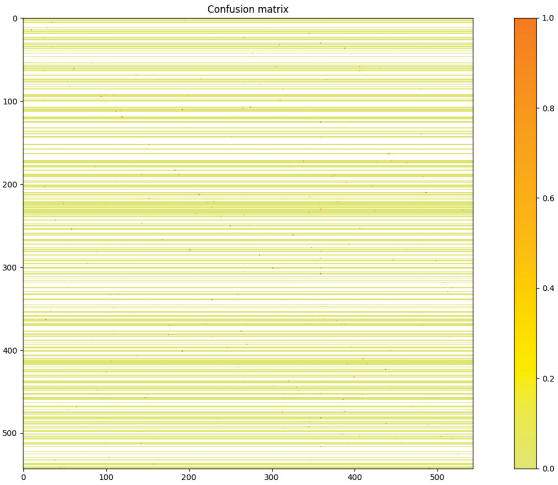


Figura A.73: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

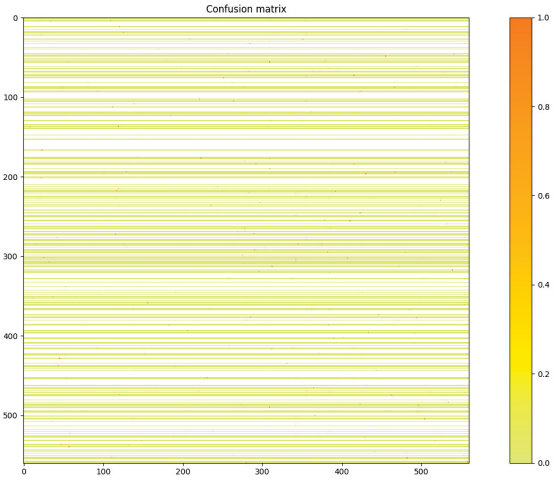


Figura A.74: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Random Forest.**

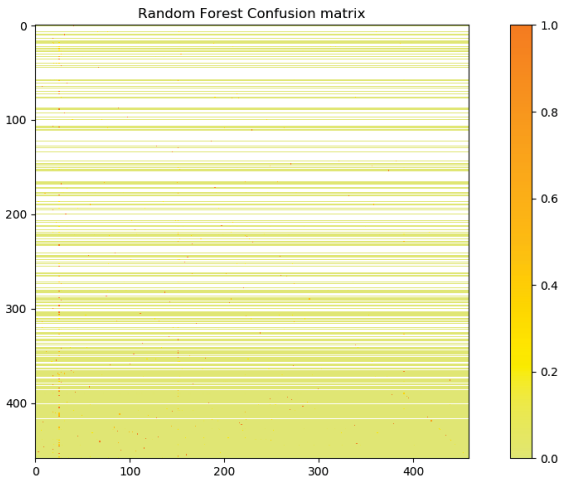


Figura A.75: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

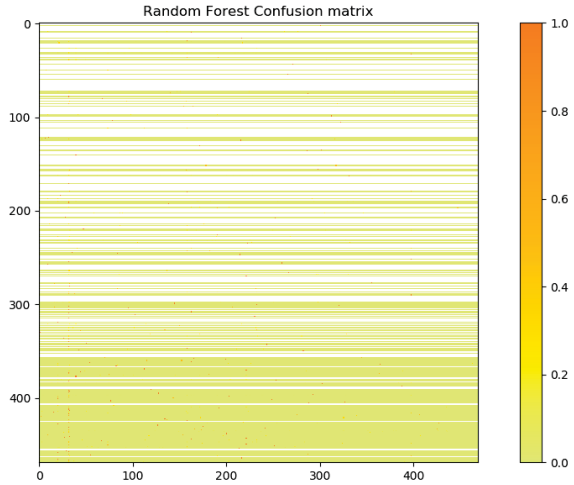


Figura A.76: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

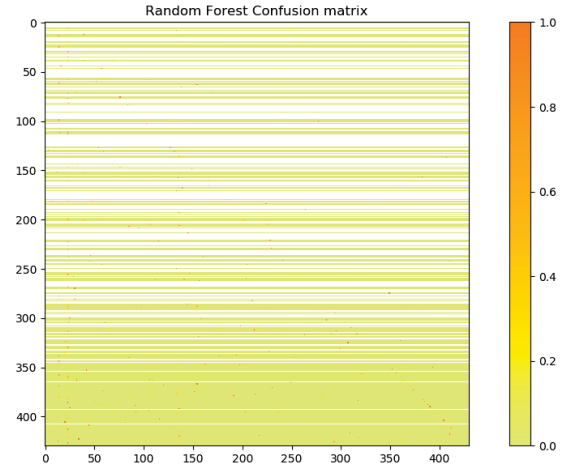


Figura A.77: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

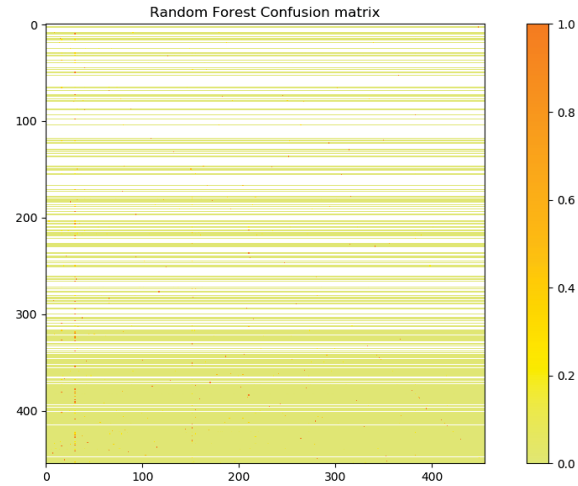


Figura A.78: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Nearest Centroid**

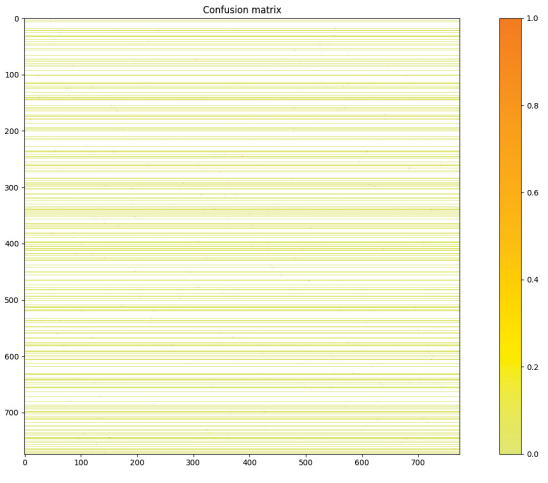


Figura A.79: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

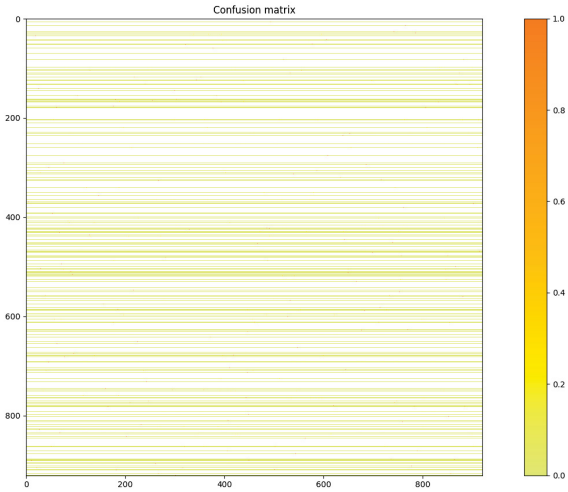


Figura A.80: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

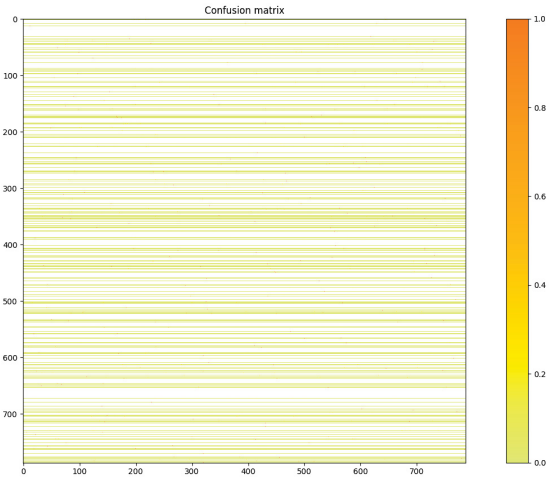


Figura A.81: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

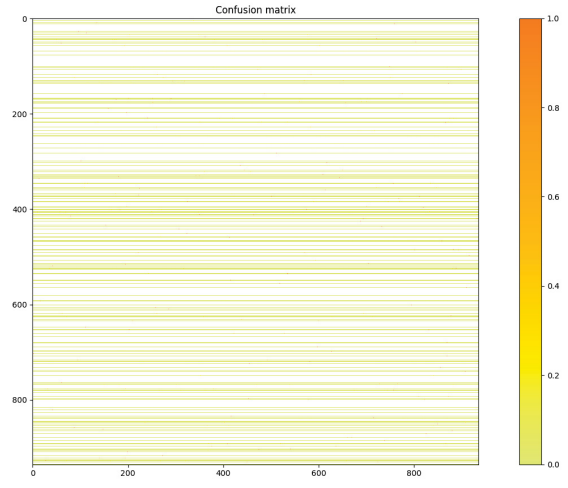


Figura A.82: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SVM.**

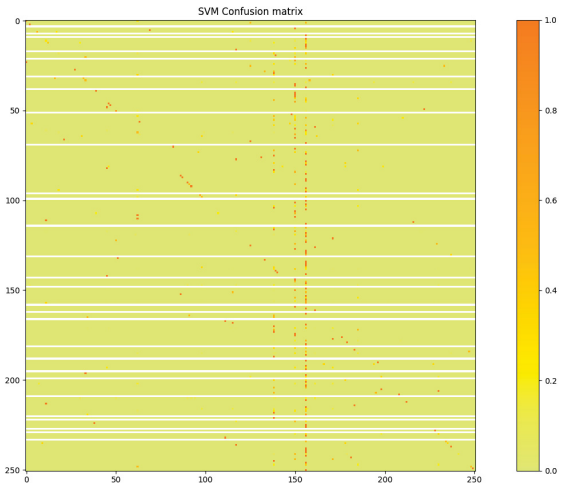


Figura A.83: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

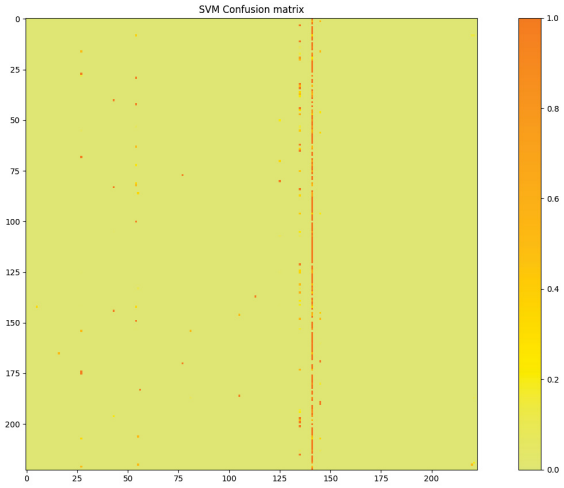


Figura A.84: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

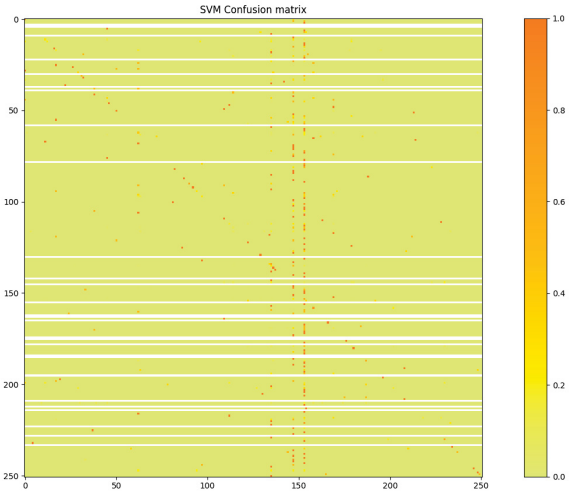


Figura A.85: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

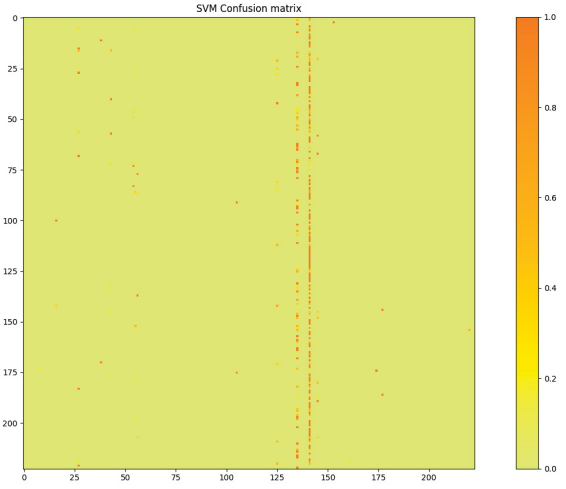


Figura A.86: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SGD.**

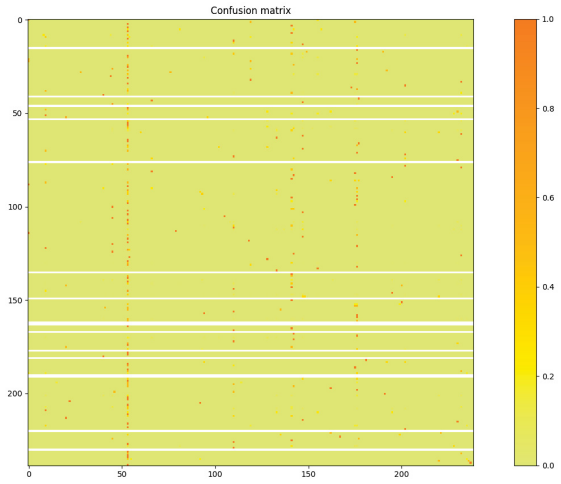


Figura A.87: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

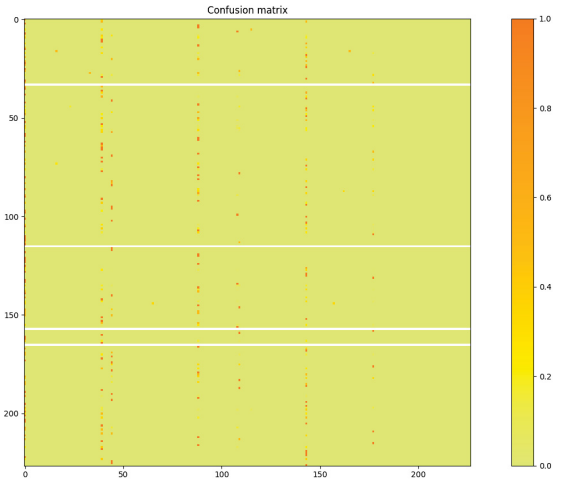


Figura A.88: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

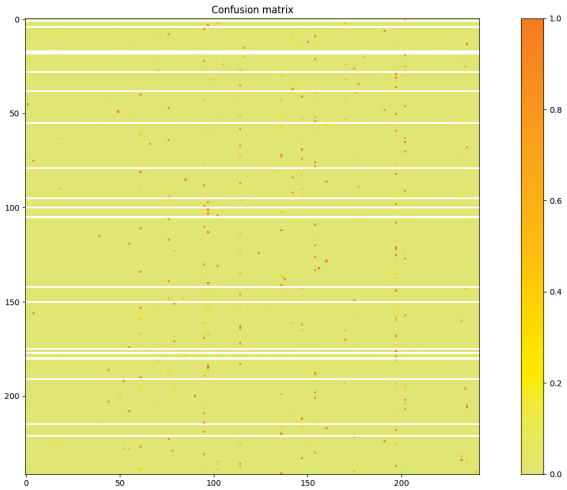


Figura A.89: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

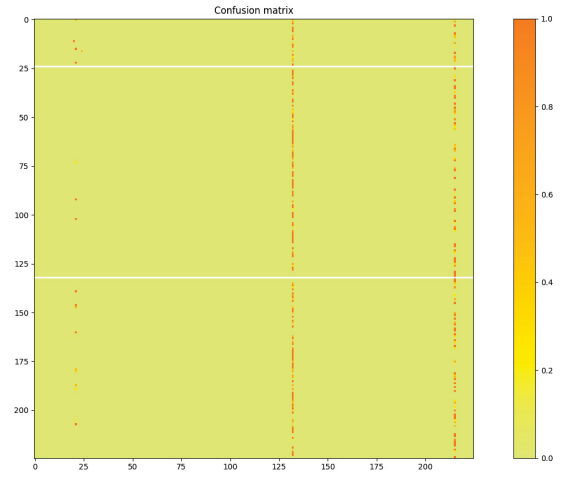


Figura A.90: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Perceptron.**

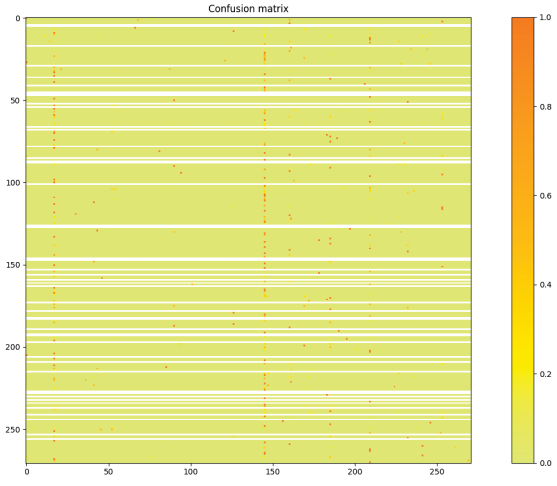


Figura A.91: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

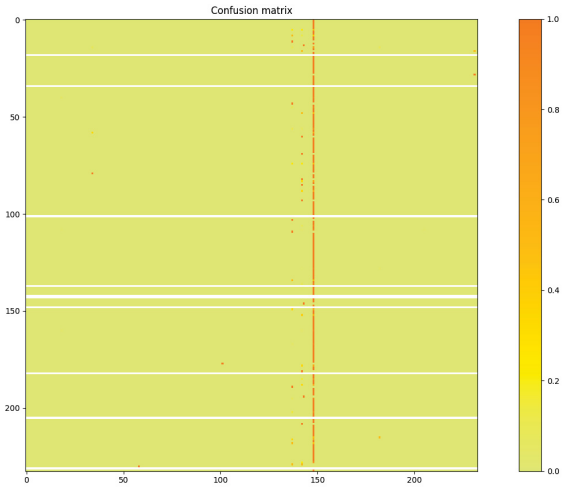


Figura A.92: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

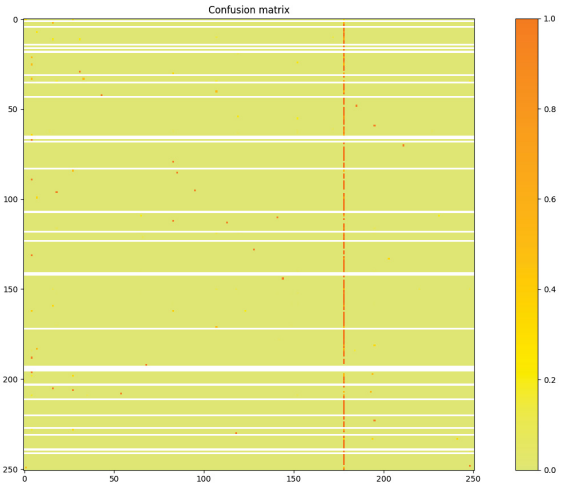


Figura A.93: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

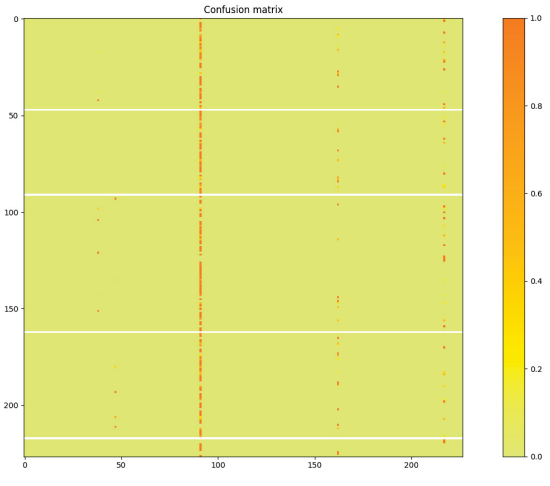


Figura A.94: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **MLP.**

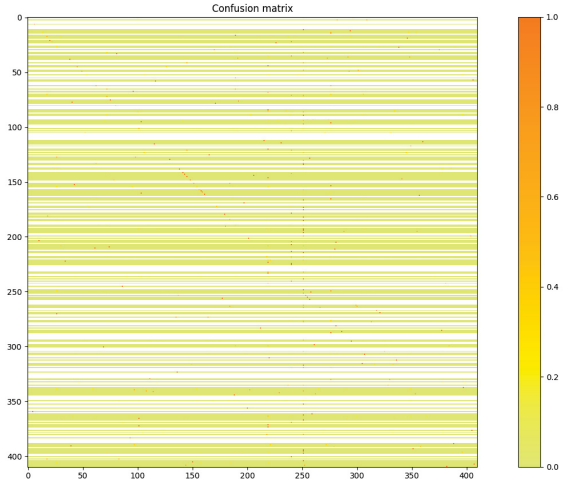


Figura A.95: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

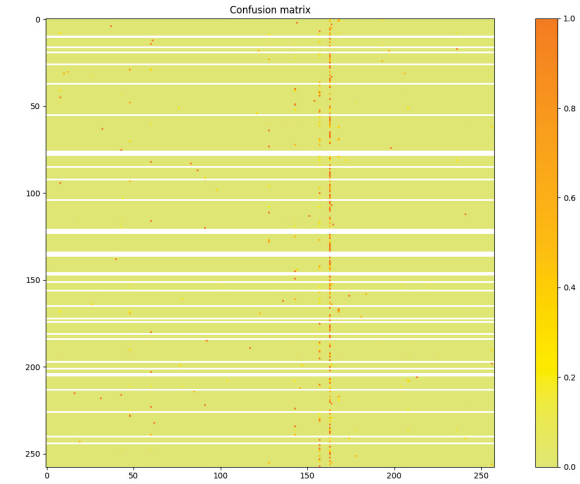


Figura A.96: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

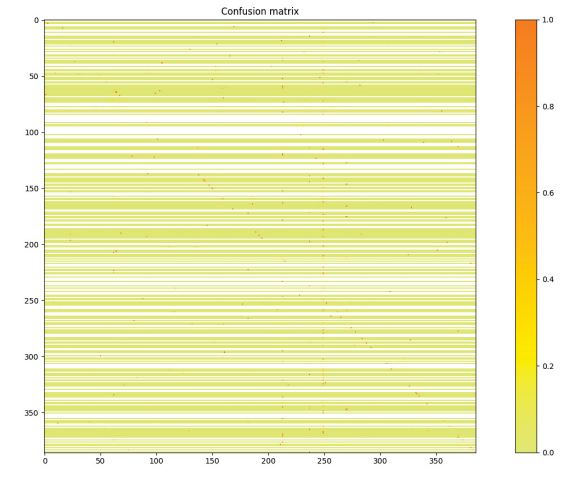


Figura A.97: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

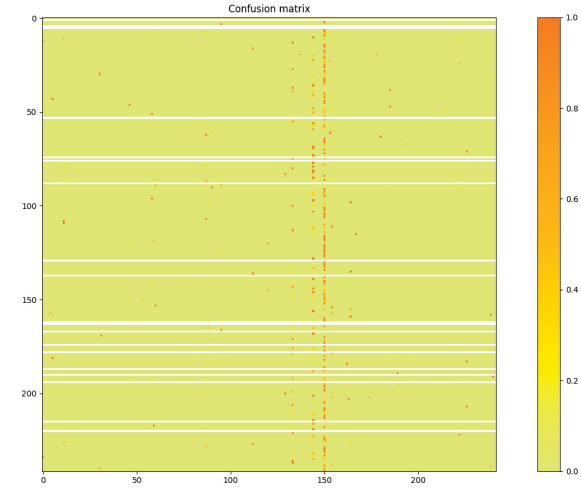


Figura A.98: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

Pela quantidade de famílias quando avaliado o *dataset* completo, não é possível observar o desempenho de cada uma delas, mas através dessas matrizes é possível notar que o descritor global normalmente exibe uma diagonal levemente marcada, o que é mais difícil de identificar nas matrizes com uso do descritor local.

A seguir foi feita a primeira seleção de dados neste *dataset*, considerando apenas famílias com no mínimo 50 exemplares. Abaixo são exibidas as matrizes de confusão dos experimentos realizados nesse subconjunto.

- **KNN.**

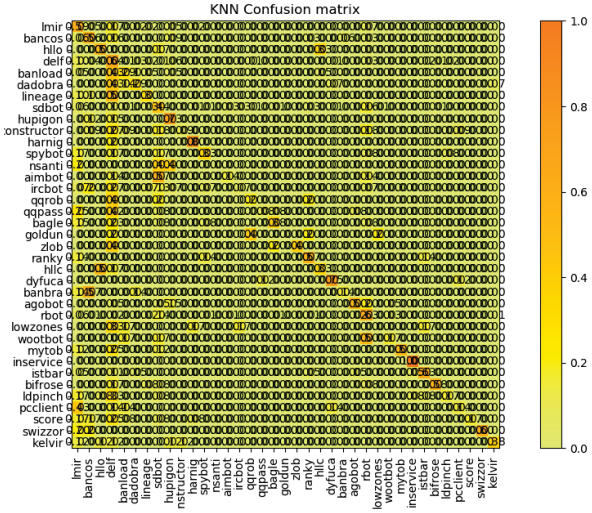


Figura A.99: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

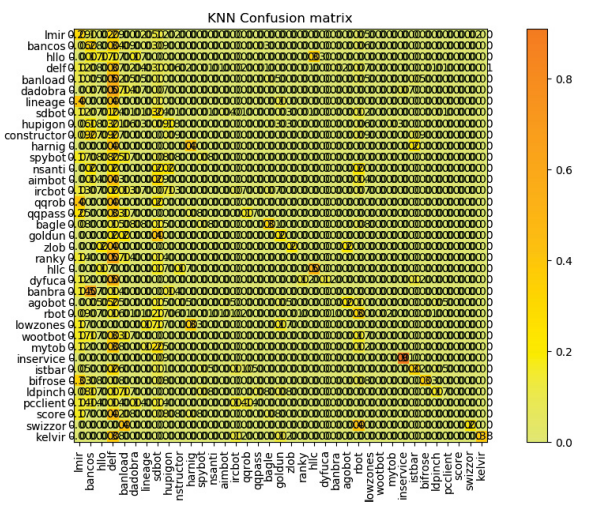


Figura A.100: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

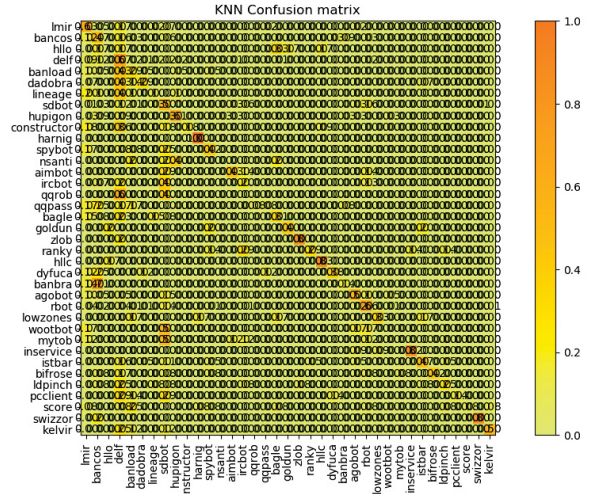


Figura A.101: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

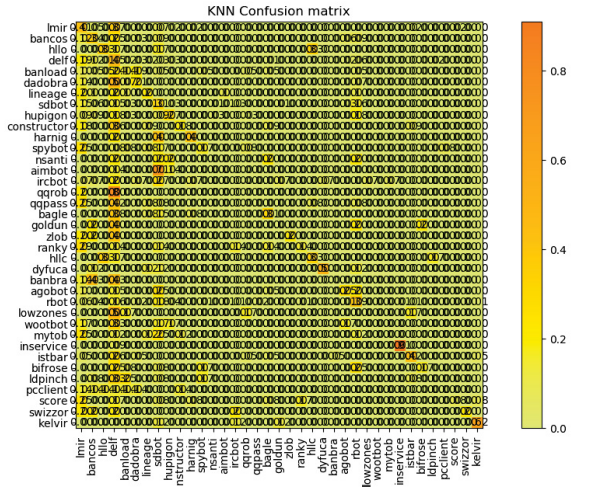


Figura A.102: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Decision Trees.**

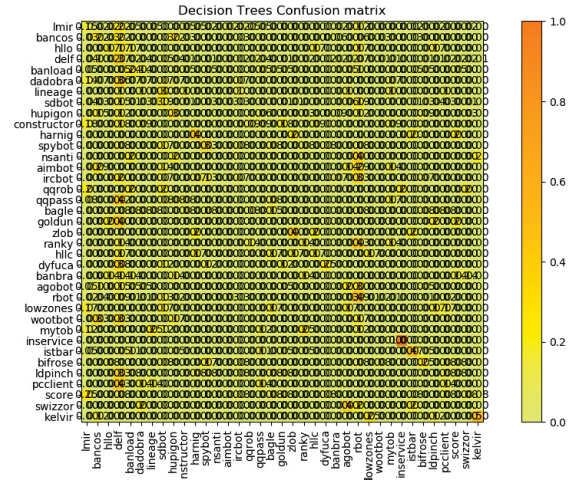


Figura A.103: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

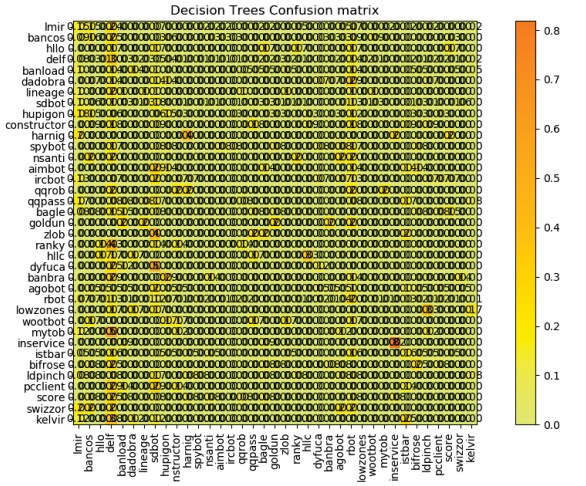


Figura A.104: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

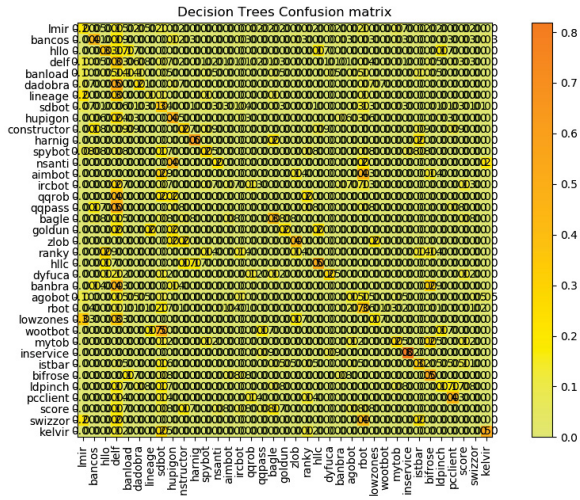


Figura A.105: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

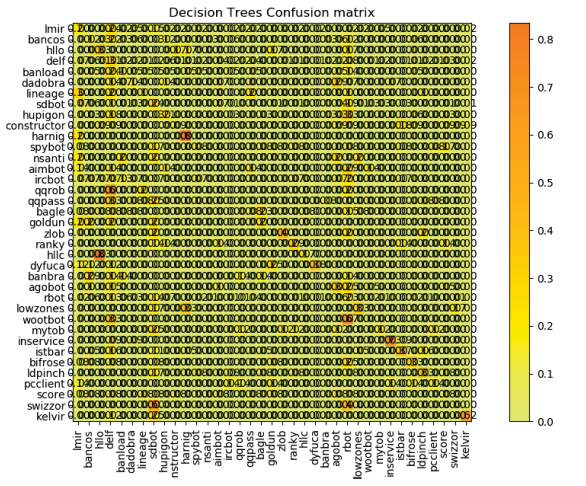


Figura A.106: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Random Forest.**

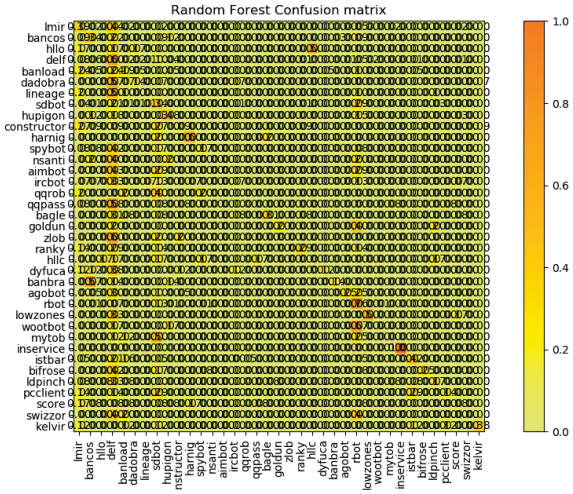


Figura A.107: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

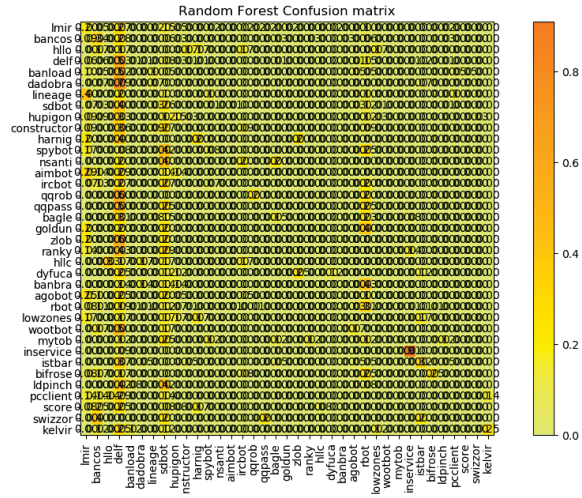


Figura A.108: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

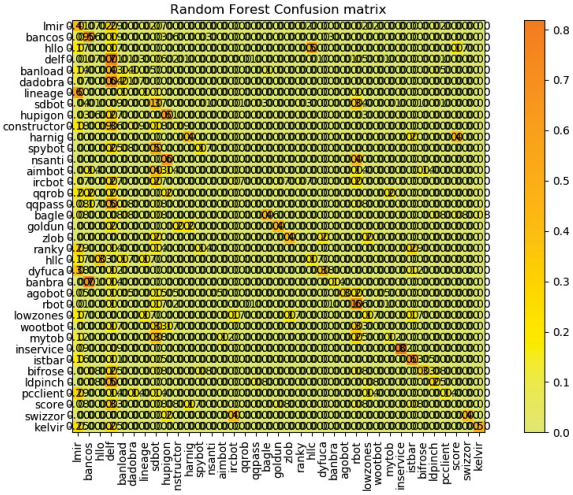


Figura A.109: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

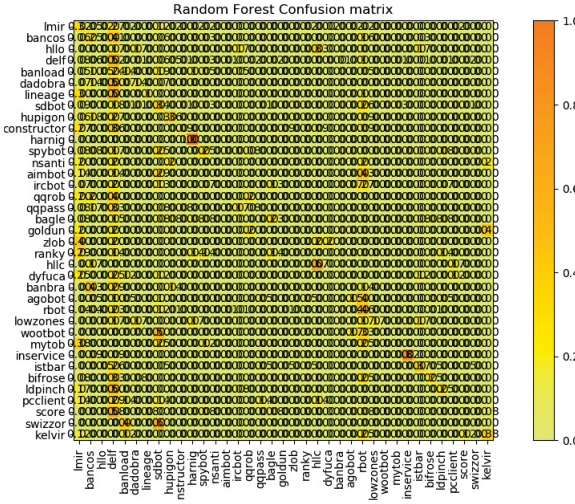


Figura A.110: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Nearest Centroid**

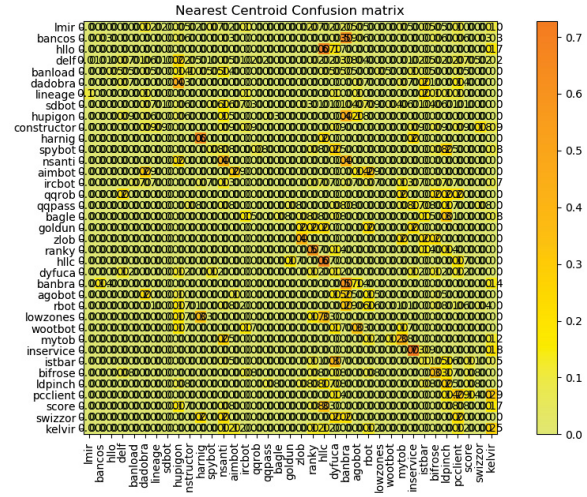


Figura A.111: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

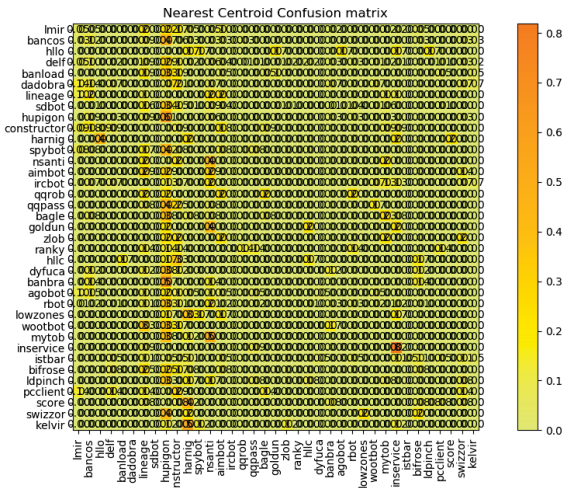


Figura A.112: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

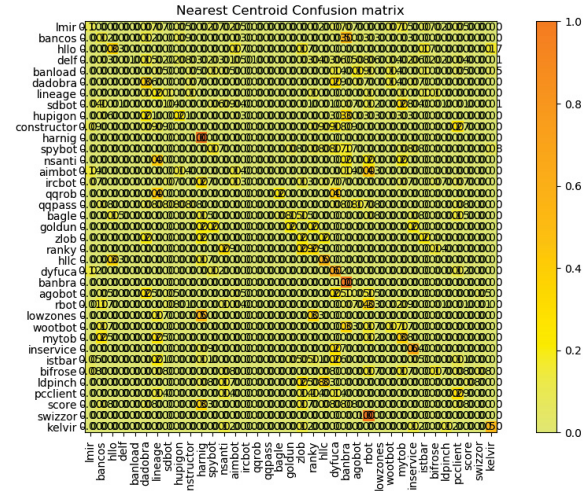


Figura A.113: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

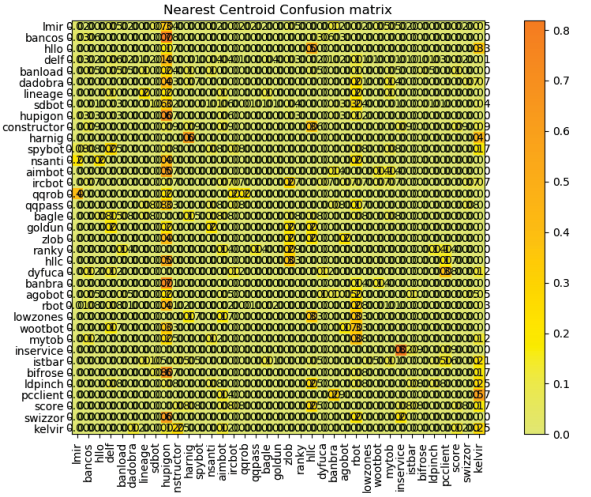


Figura A.114: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SVM.**

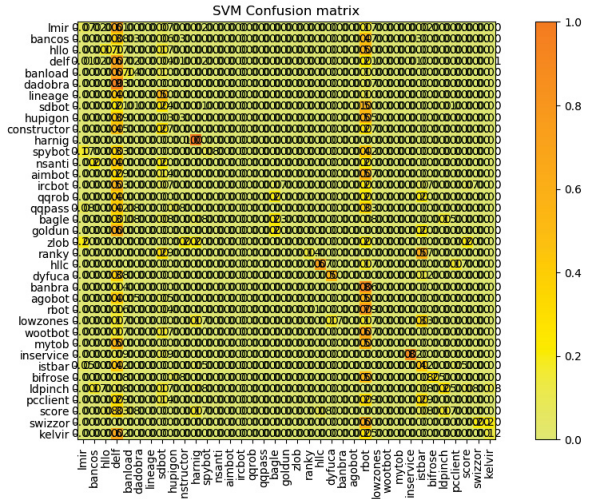


Figura A.115: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

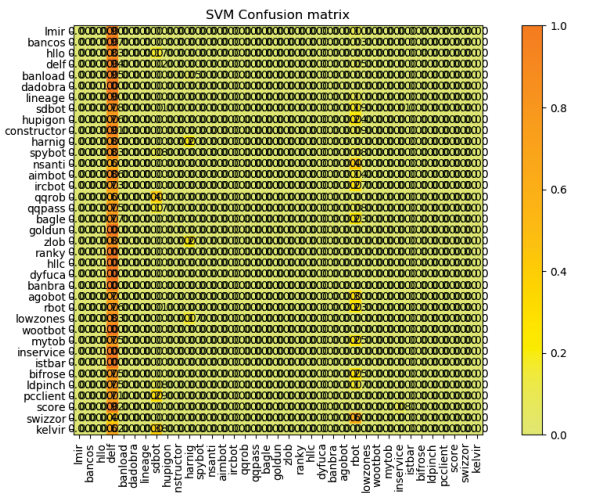


Figura A.116: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

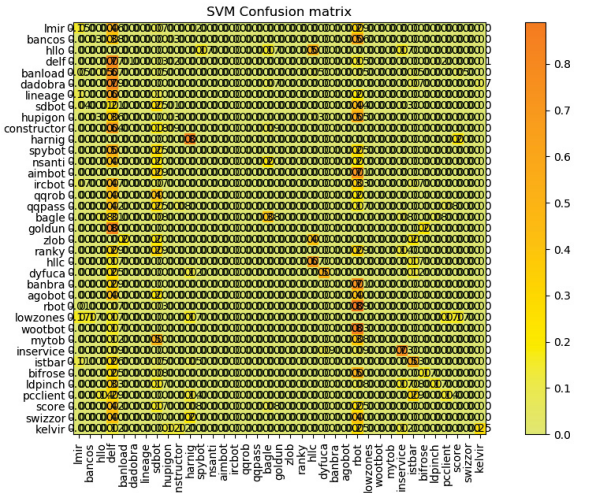


Figura A.117: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

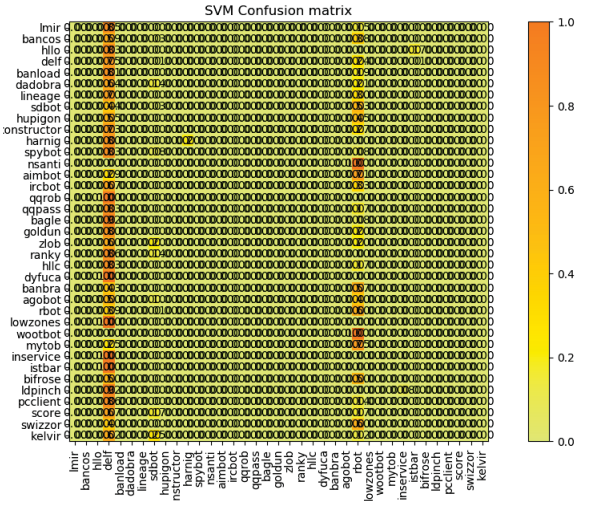


Figura A.118: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SGD.**

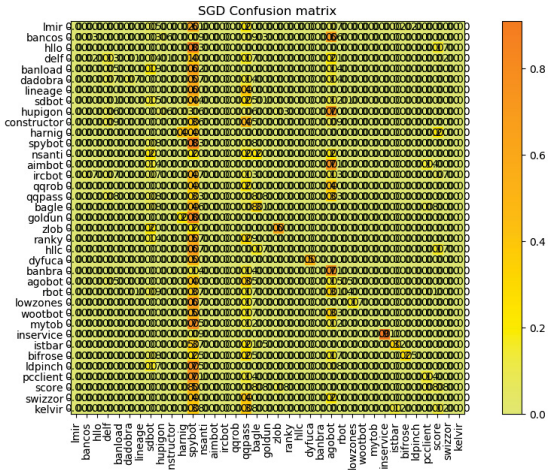


Figura A.119: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

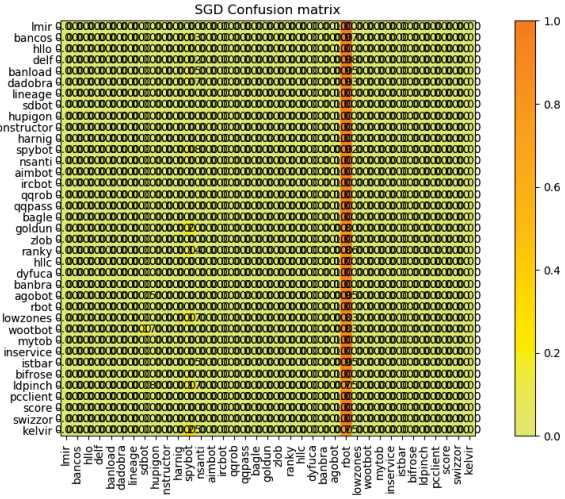


Figura A.120: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

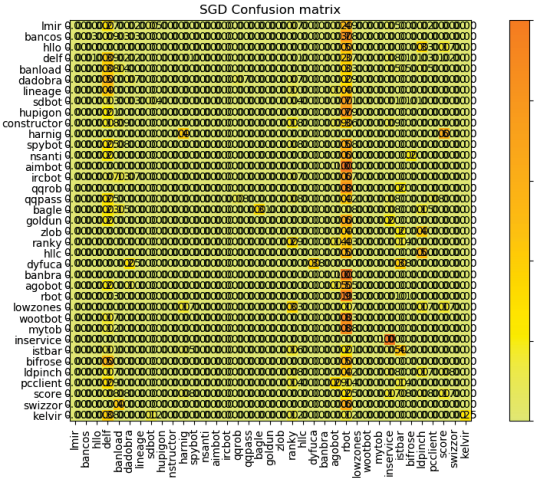


Figura A.121: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

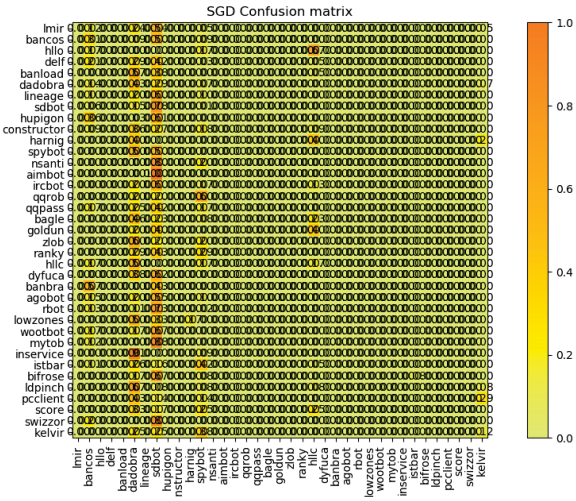


Figura A.122: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Perceptron.**

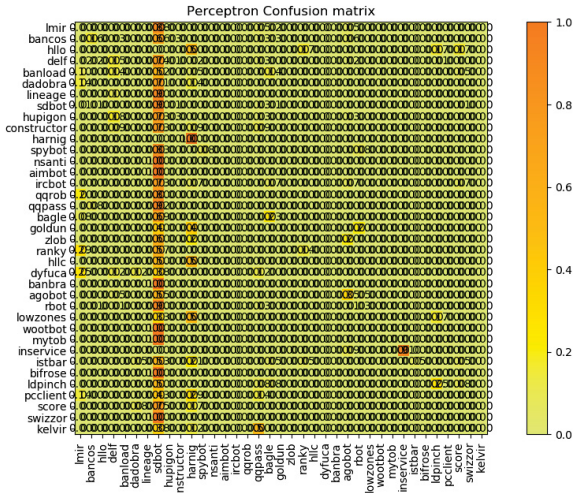


Figura A.123: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

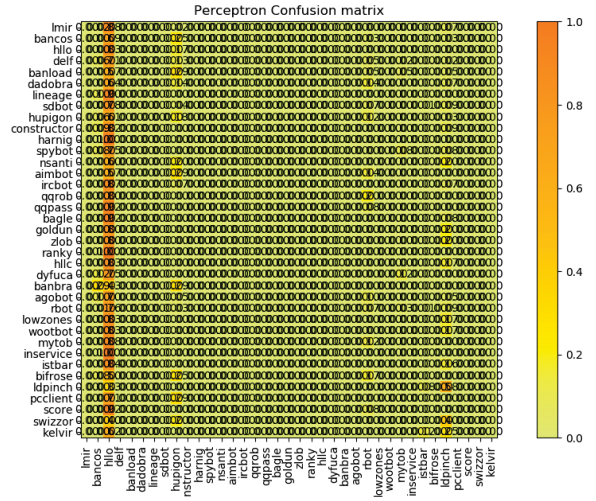


Figura A.124: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

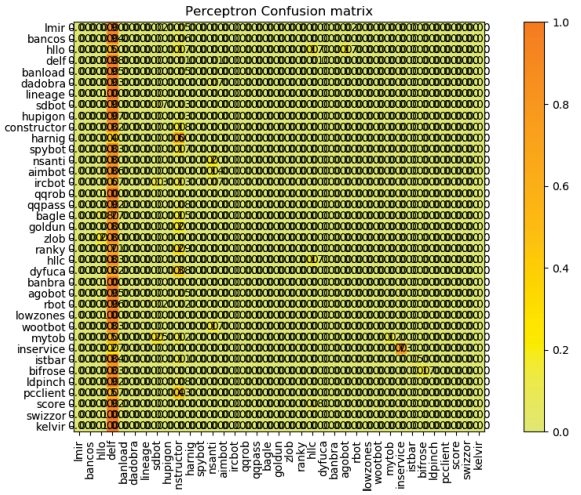


Figura A.125: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

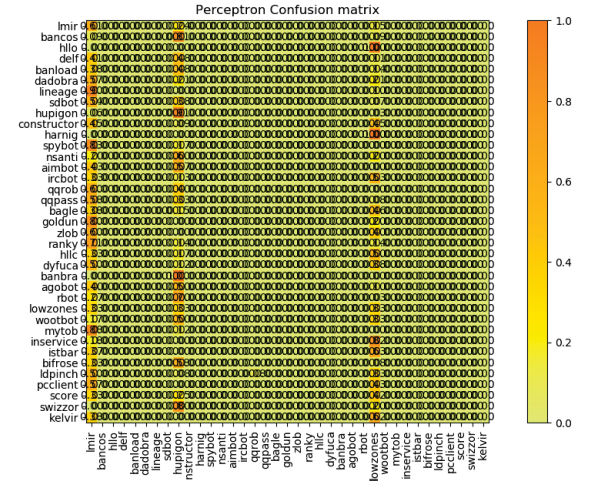


Figura A.126: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **MLP.**

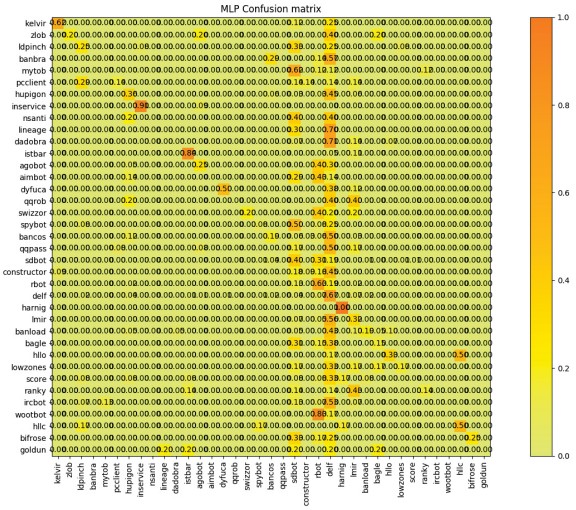


Figura A.127: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

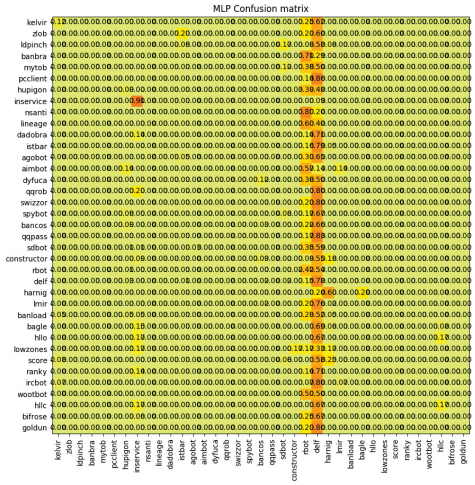


Figura A.128: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

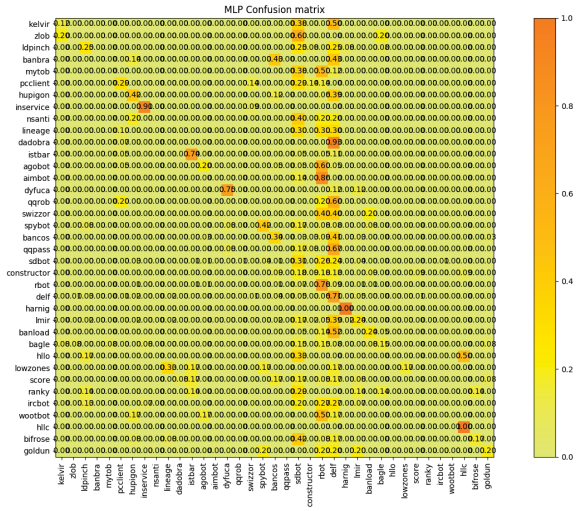


Figura A.129: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

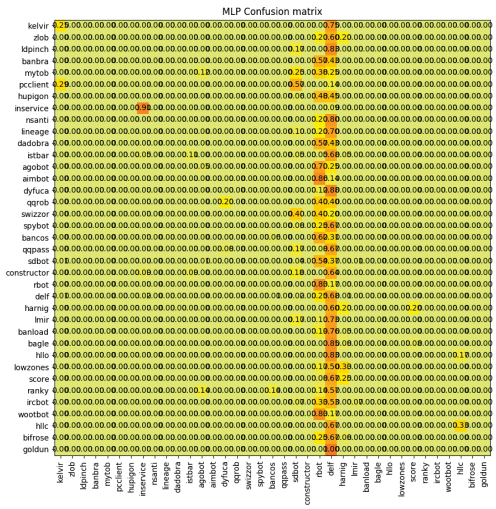


Figura A.130: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **CNN.**

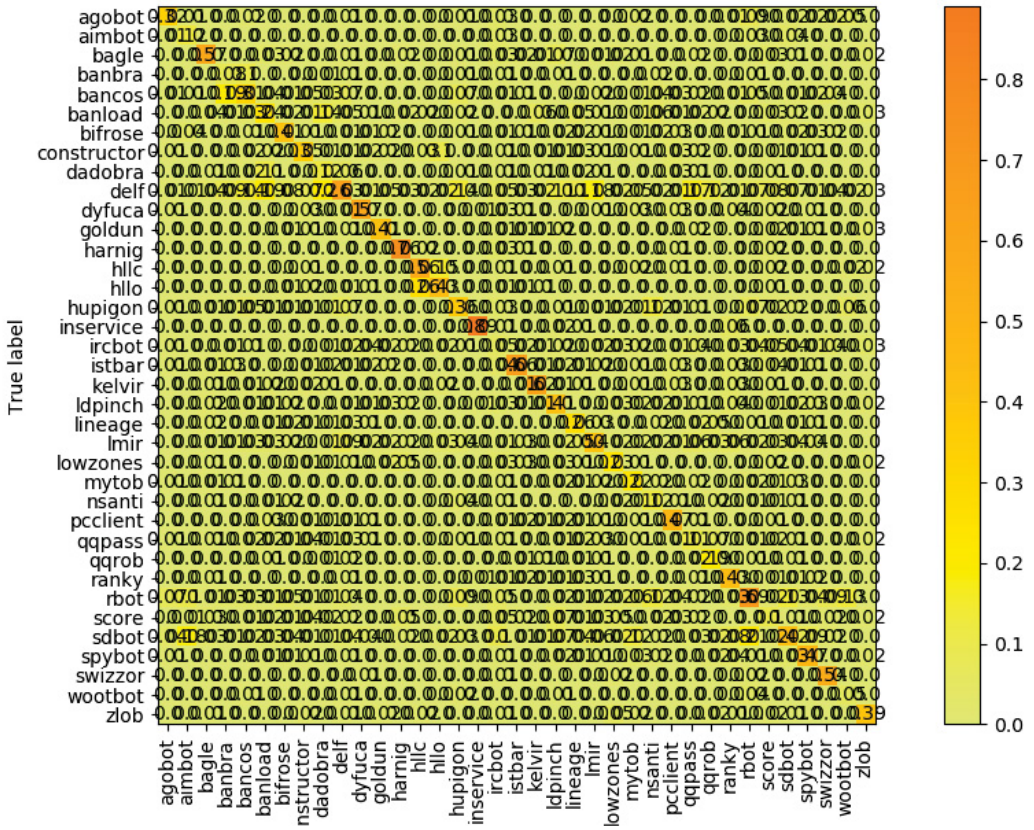


Figura A.131: Matriz de confusão do classificador CNN no subconjunto do Malimg Dataset.

Nesses experimentos é possível notar o impacto do uso de dados não balanceados na classificação. Identifica-se que muitas amostras são confundidas principalmente com as famílias `Delf`, `Rbot` e `Sdbot`, que são as maiores do *dataset*.

A seguir são exibidas as matrizes de confusão geradas pelos classificadores aplicados no subconjunto do *dataset* local com as 10 famílias mais representativas. Esse subconjunto foi balanceado com 100 amostras por família para não apresentar o mesmo problema identificado no subconjunto anterior.

- **KNN.**

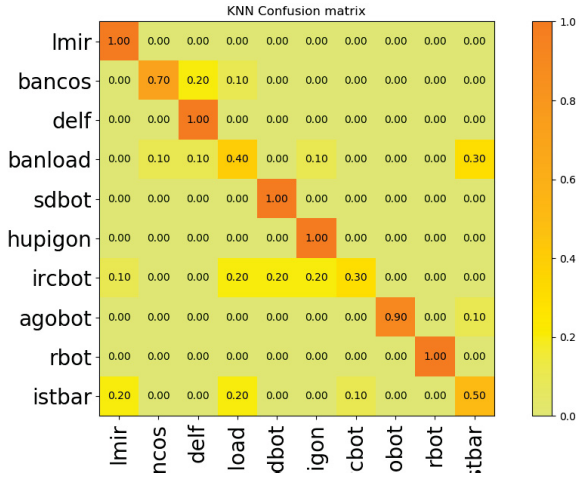


Figura A.132: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

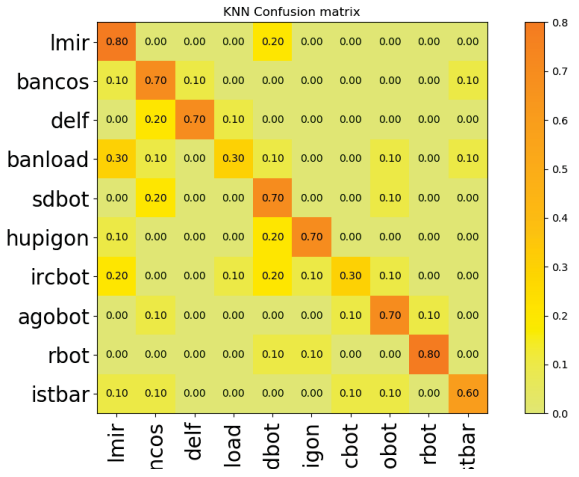


Figura A.133: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

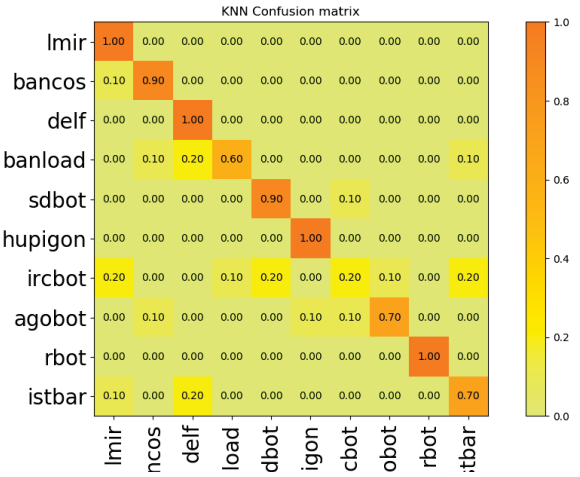


Figura A.134: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

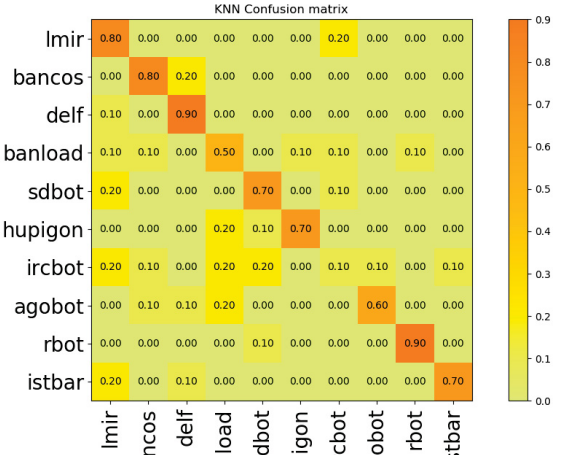


Figura A.135: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Decision Trees.**

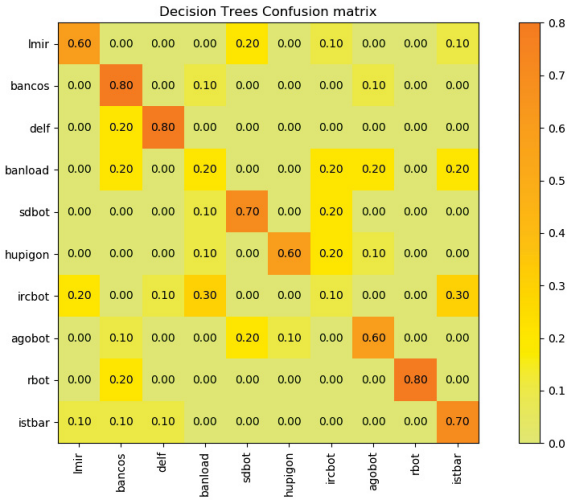


Figura A.136: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

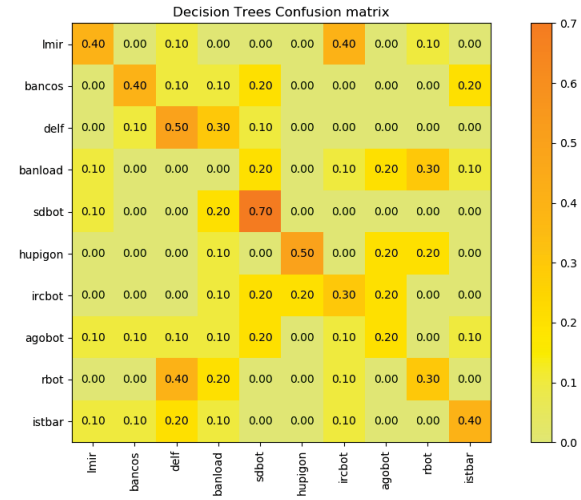


Figura A.137: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

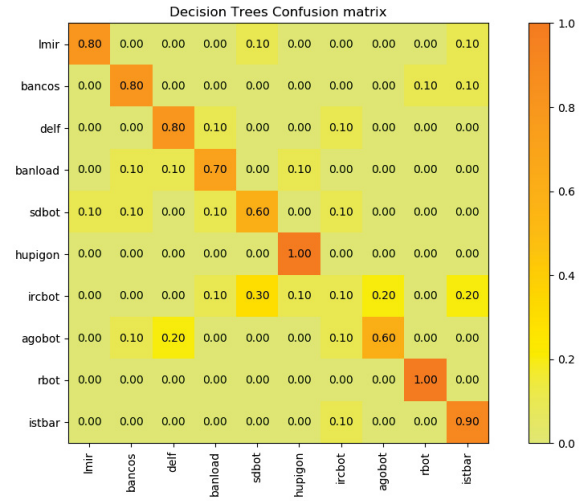


Figura A.138: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

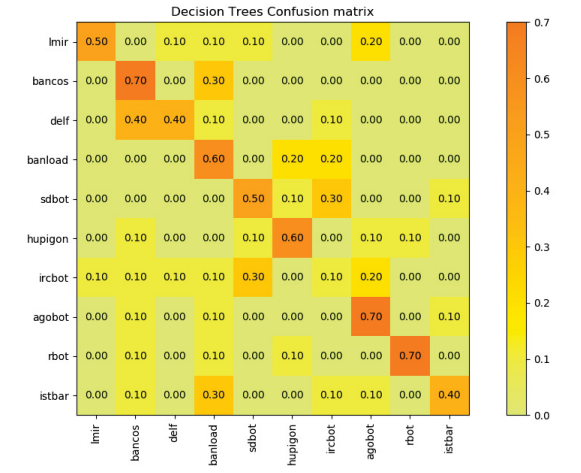


Figura A.139: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Random Forest.**

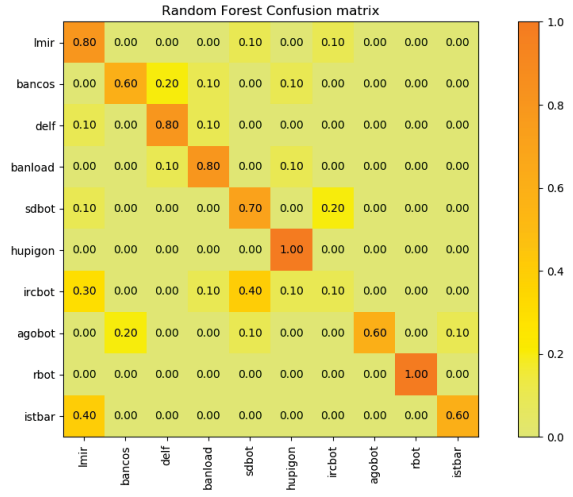


Figura A.140: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

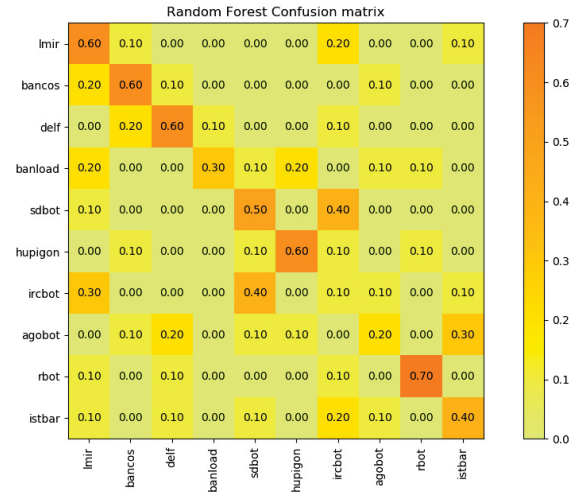


Figura A.141: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

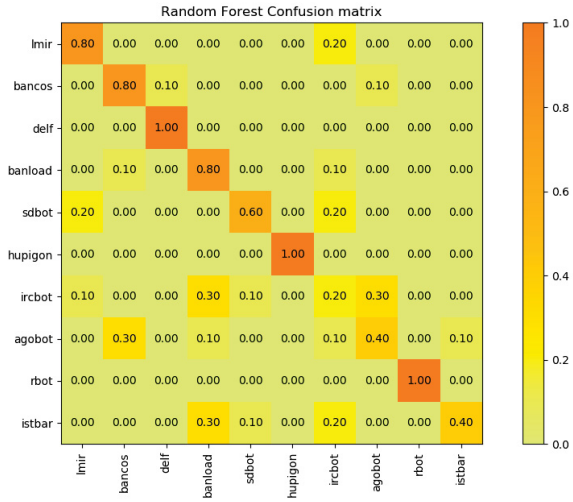


Figura A.142: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

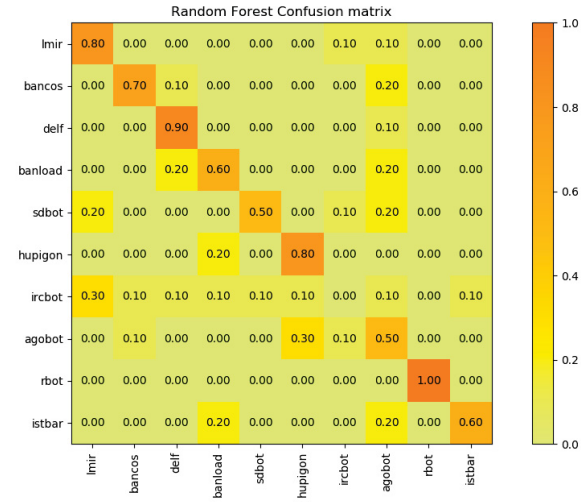


Figura A.143: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Nearest Centroid**

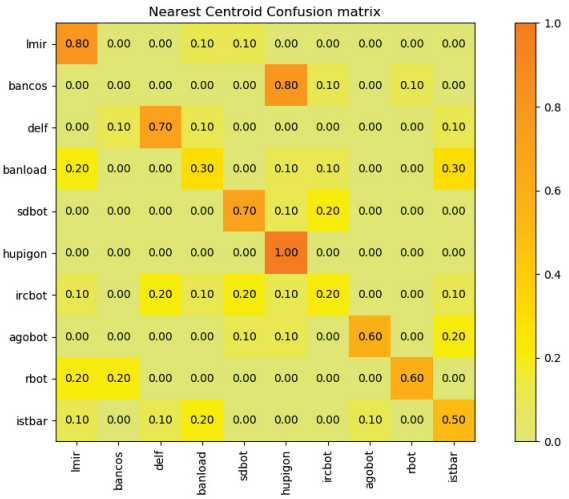


Figura A.144: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

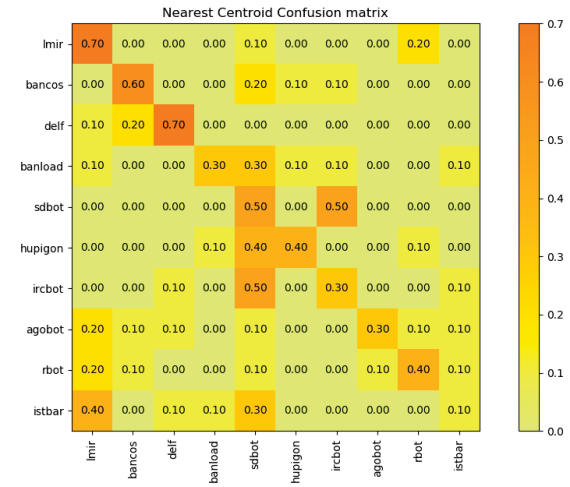


Figura A.145: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

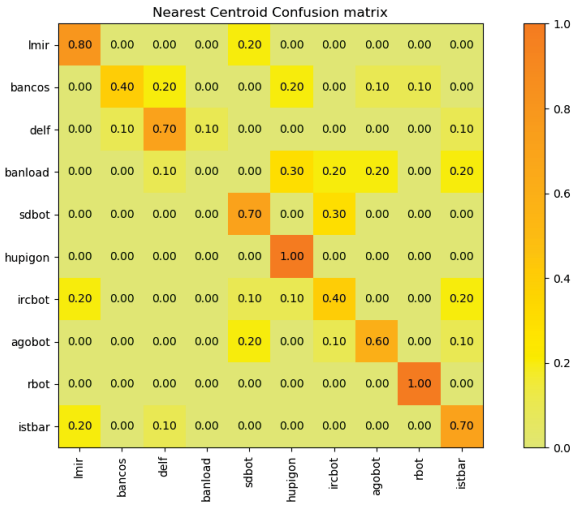


Figura A.146: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

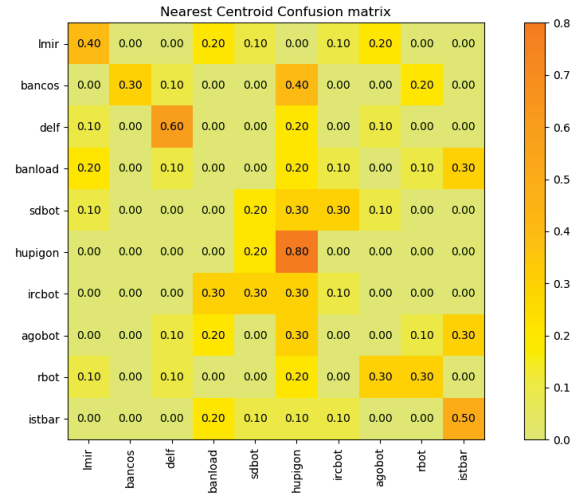


Figura A.147: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SVM.**

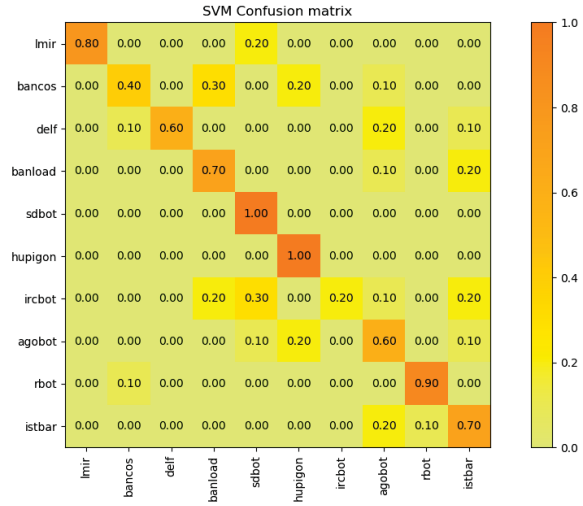


Figura A.148: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

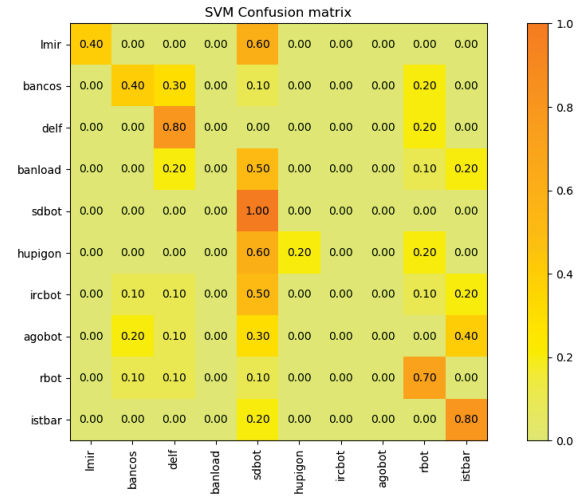


Figura A.149: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

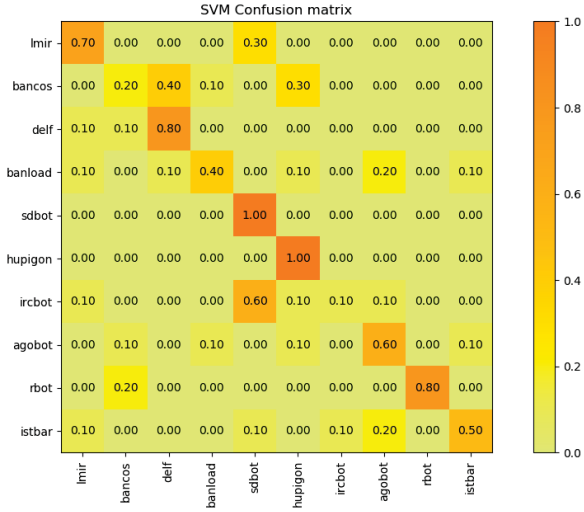


Figura A.150: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

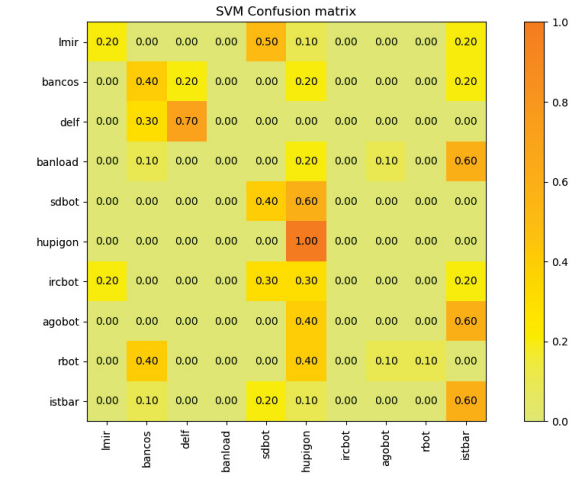


Figura A.151: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SGD.**

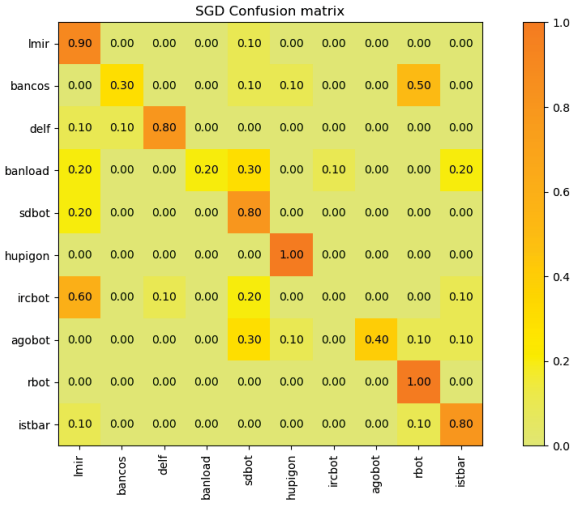


Figura A.152: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

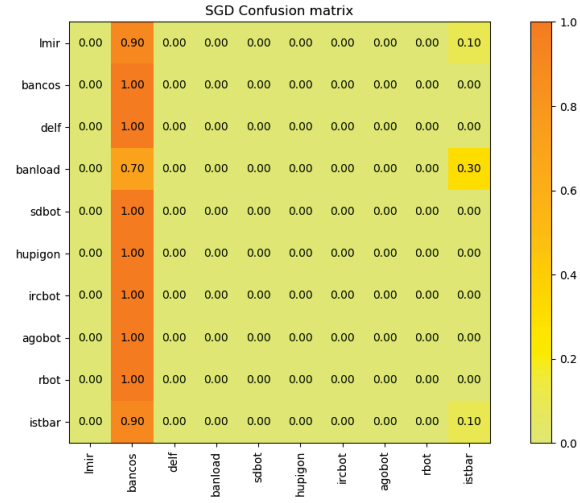


Figura A.153: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

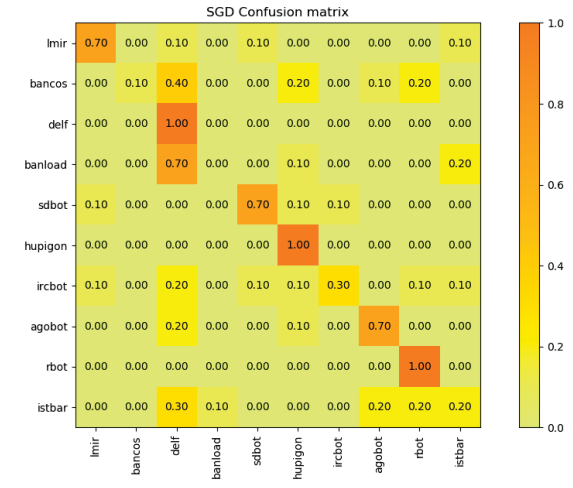


Figura A.154: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

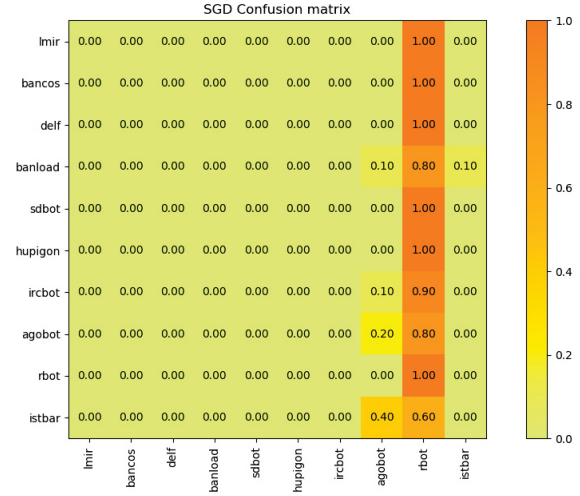


Figura A.155: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Perceptron.**

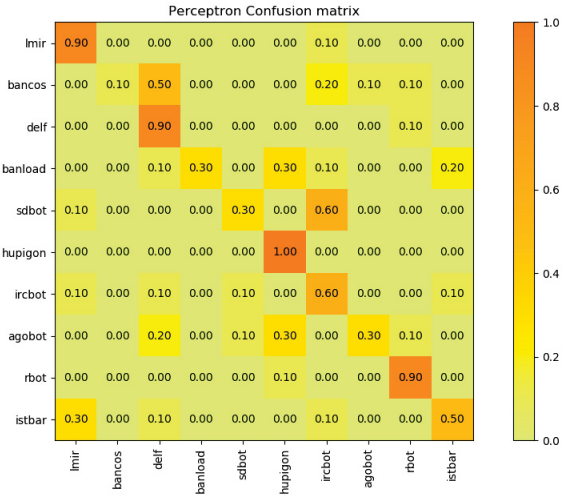


Figura A.156: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

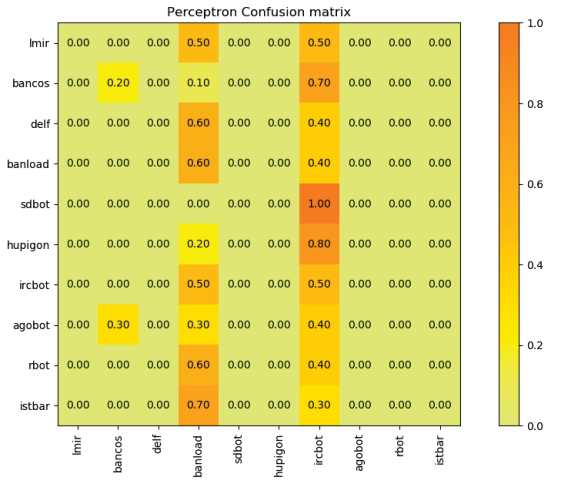


Figura A.157: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

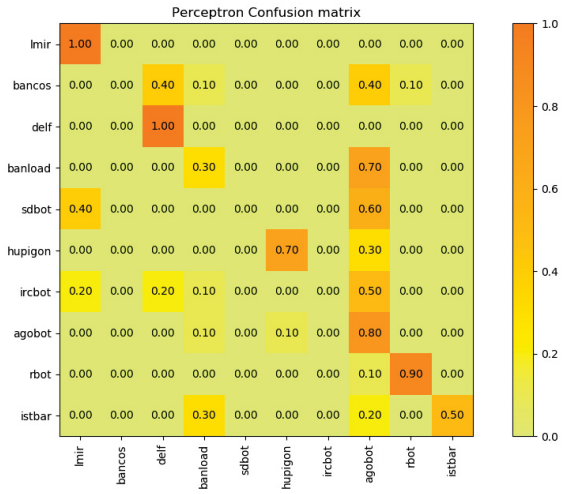


Figura A.158: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

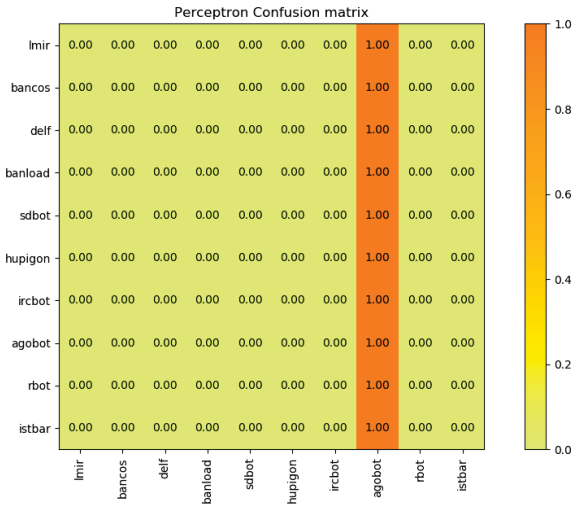


Figura A.159: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **MLP.**

Figura A.160: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

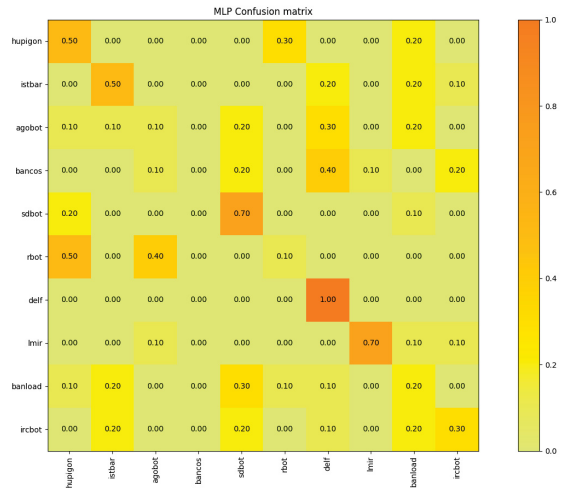


Figura A.161: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

Figura A.162: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

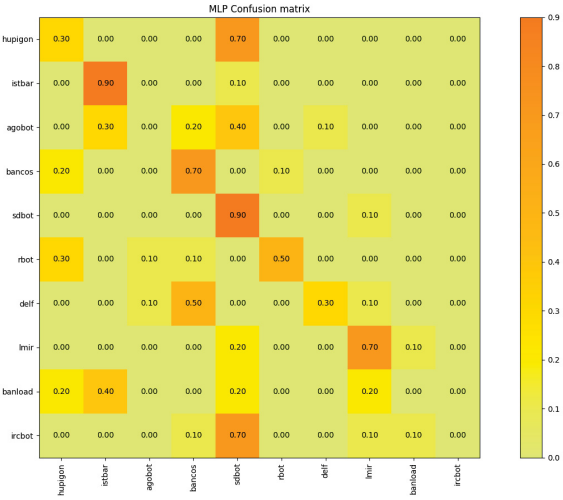


Figura A.163: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **CNN.**

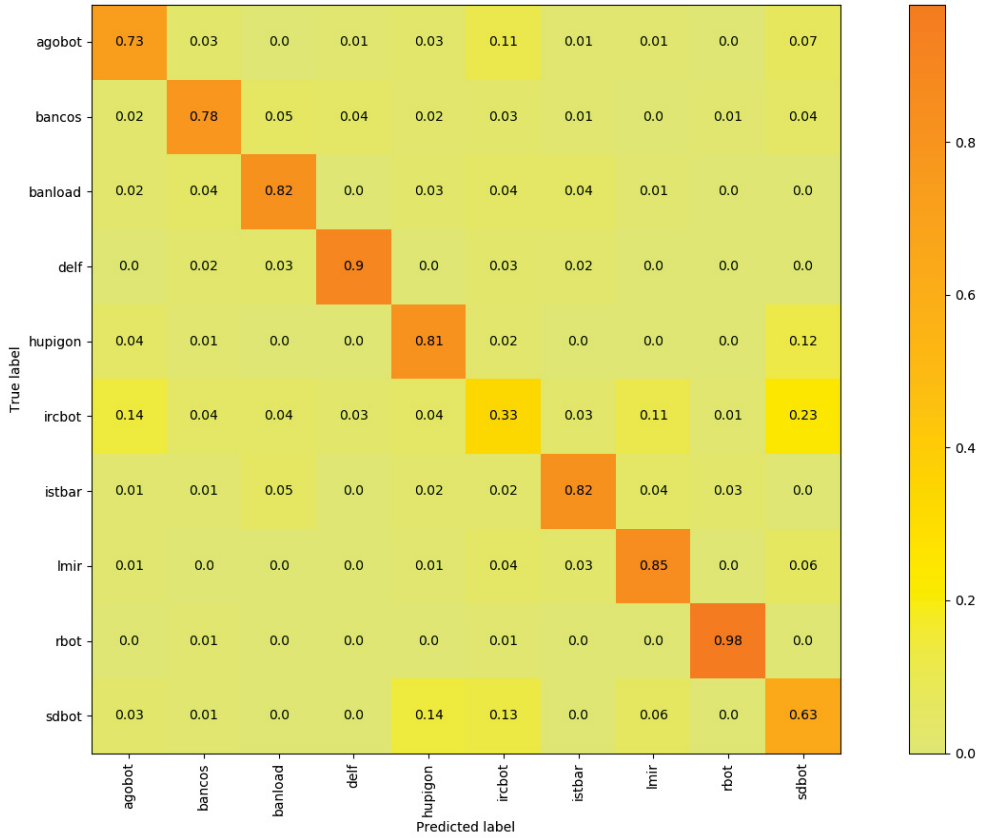


Figura A.164: Matriz de confusão do classificador CNN no subconjunto do Malimg Dataset.

Novamente são identificadas algumas famílias que possuem uma taxa de classificação inferior na maioria dos experimentos. As famílias `Ircbot`, `Banload` e `Agobot` são as que apresentam os piores resultados. Apesar disso, para a nova seleção de dados foi continuado o critério de seleção das famílias mais significativas, excluindo assim a família `Ircbot` e a família `Istbar`. As matrizes de confusão do subconjunto de 8 famílias do *dataset* local são apresentadas nas Figuras abaixo.

- **KNN.**

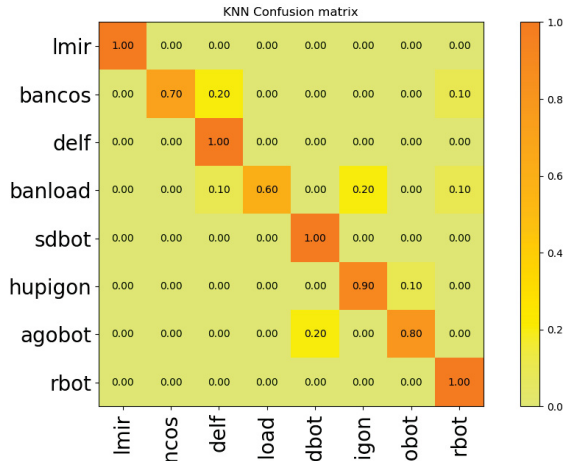


Figura A.165: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

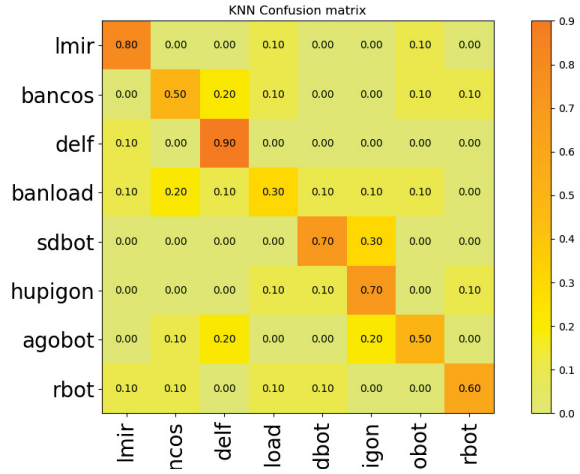


Figura A.166: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

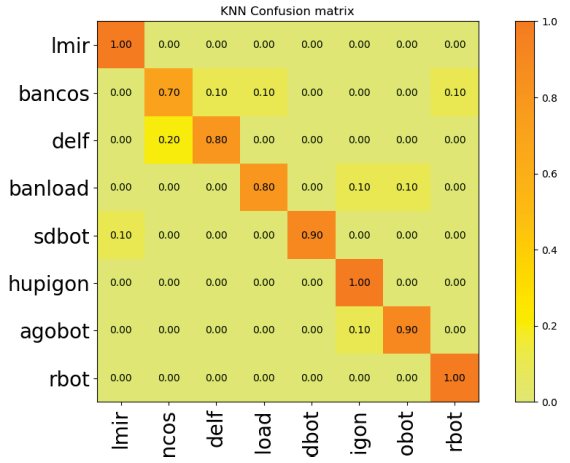


Figura A.167: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

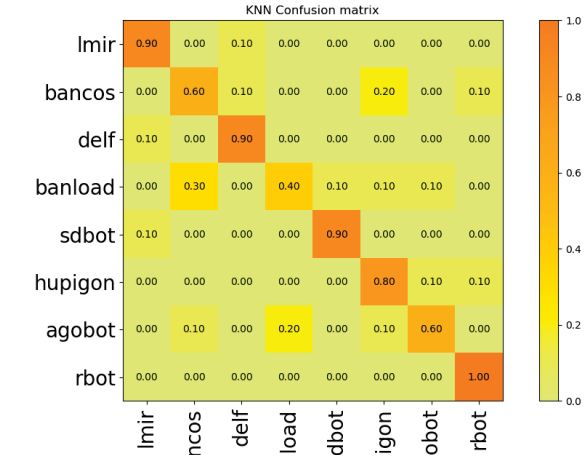


Figura A.168: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Decision Trees.**

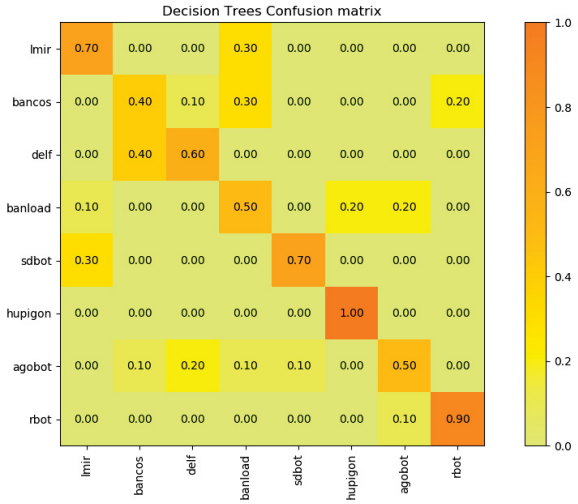


Figura A.169: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

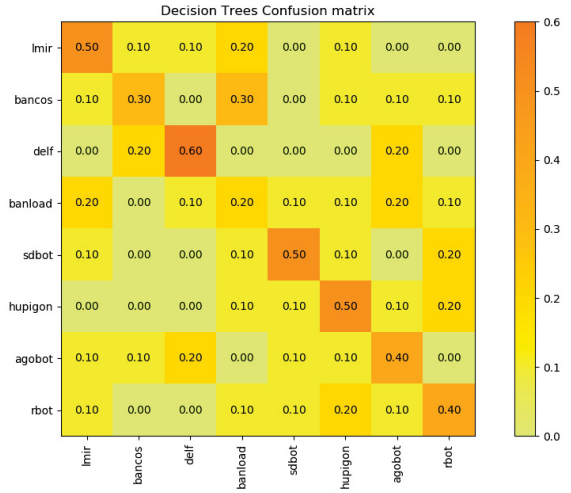


Figura A.170: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

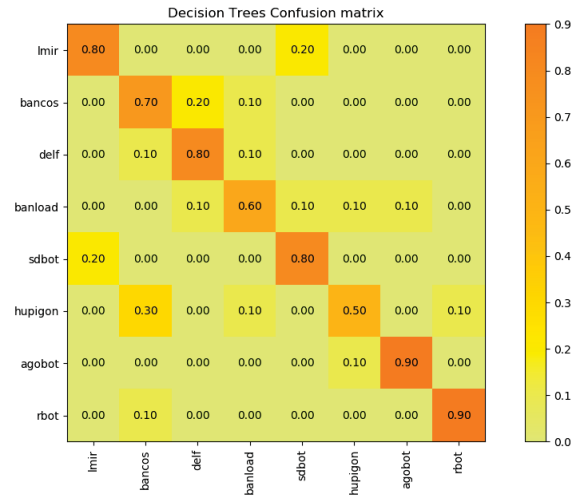


Figura A.171: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

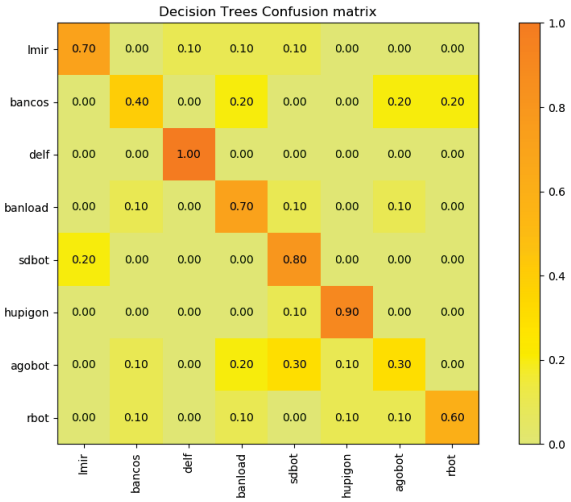


Figura A.172: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Random Forest.**

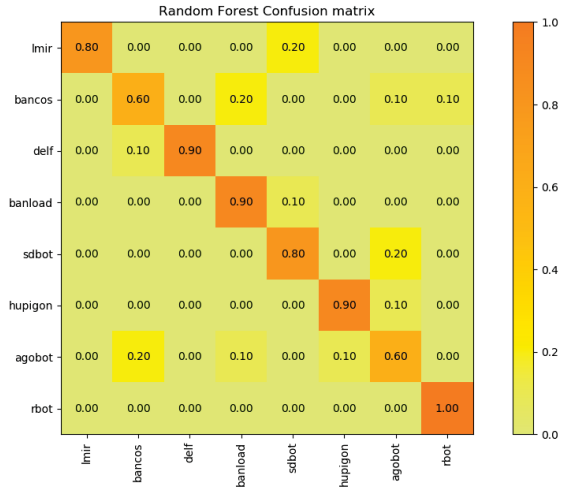


Figura A.173: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

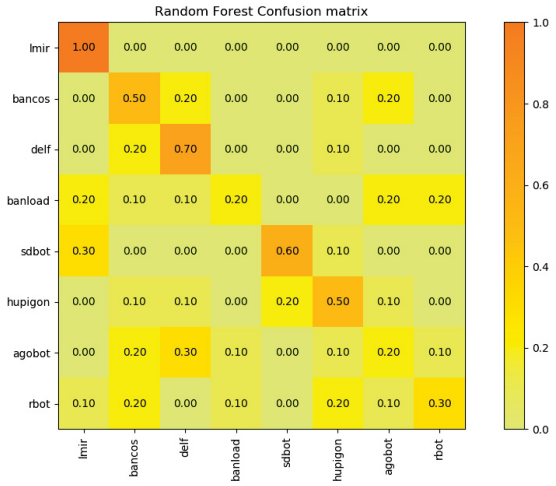


Figura A.174: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

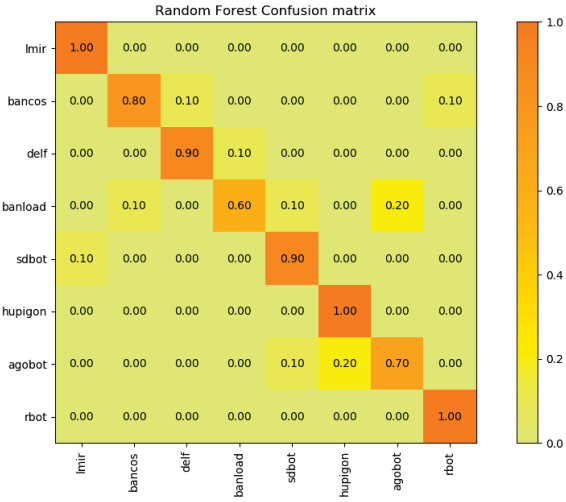


Figura A.175: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

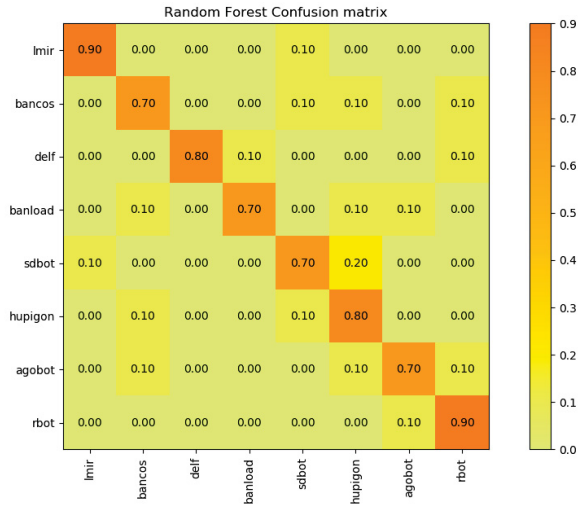


Figura A.176: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Nearest Centroid**

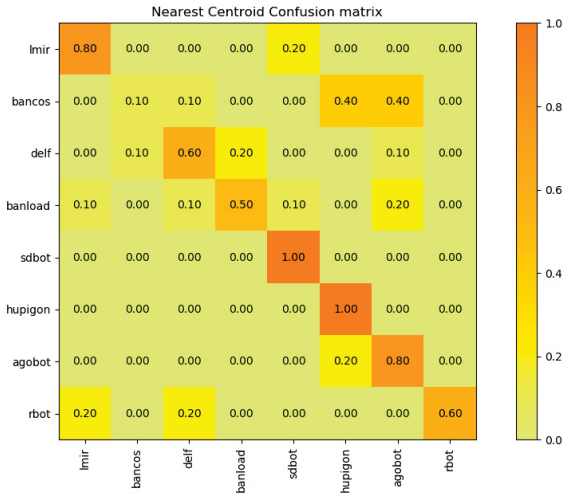


Figura A.177: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

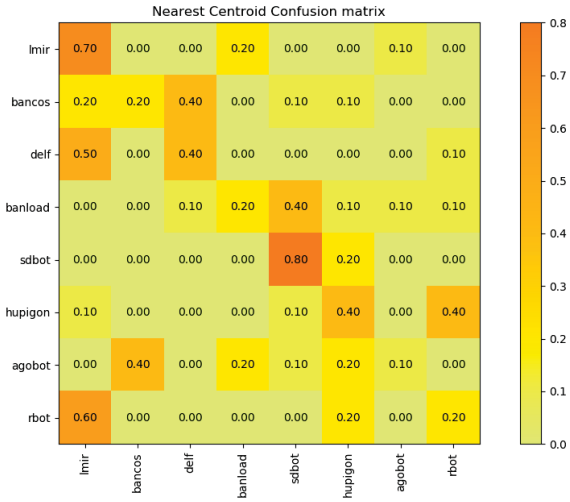


Figura A.178: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

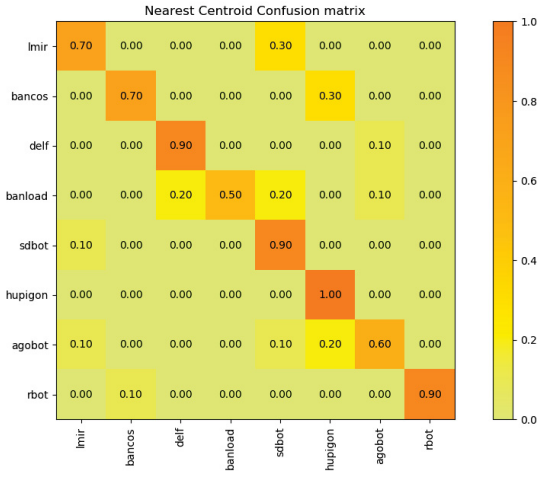


Figura A.179: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

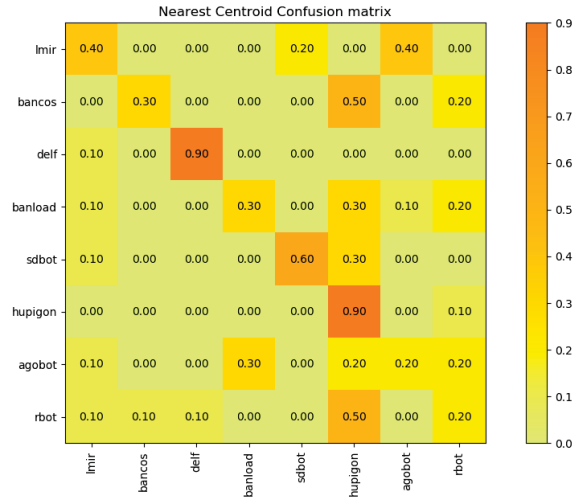


Figura A.180: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SVM.**

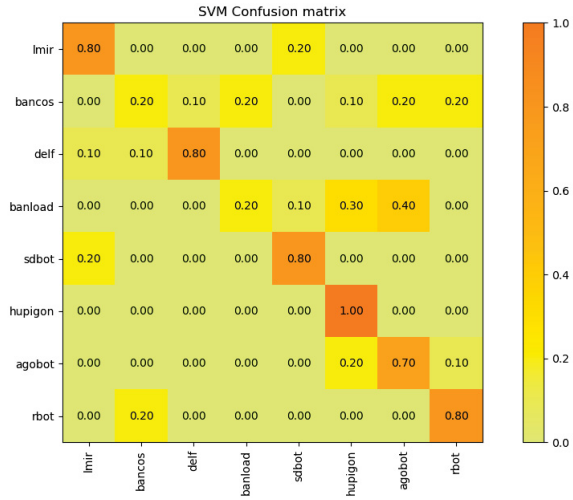


Figura A.181: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

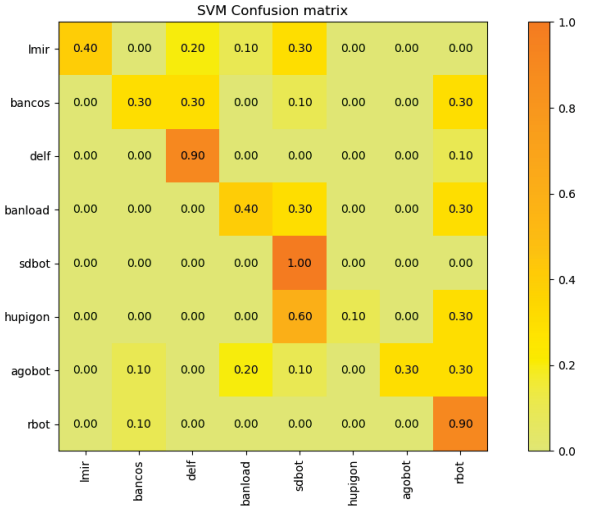


Figura A.182: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 no subconjunto do Malimg Dataset.

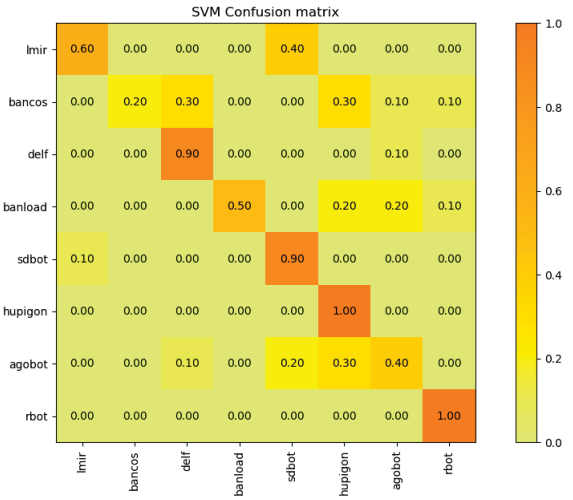


Figura A.183: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

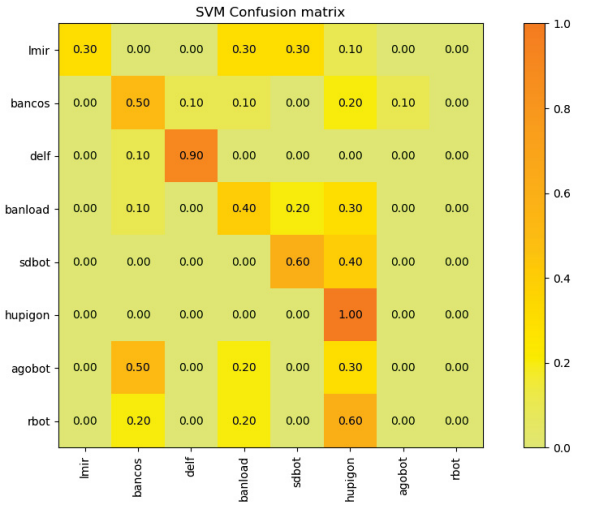


Figura A.184: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **SGD.**

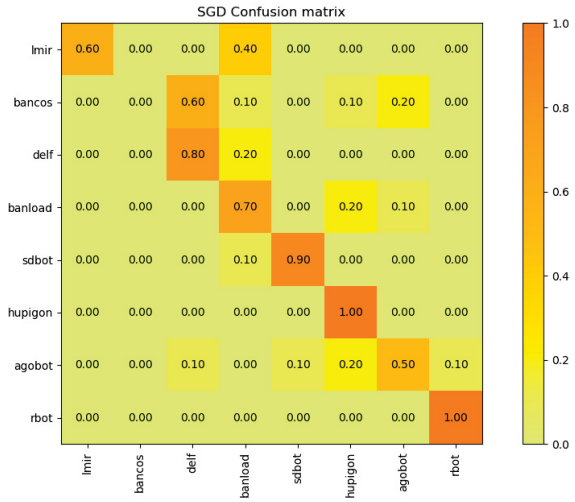


Figura A.185: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

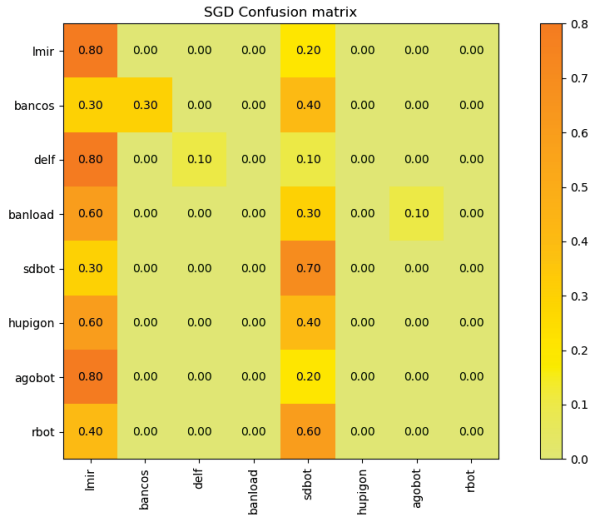


Figura A.186: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

Figura A.187: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

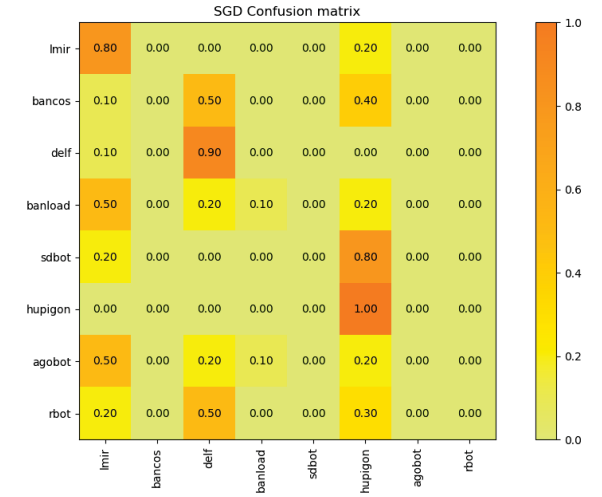


Figura A.188: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **Perceptron.**

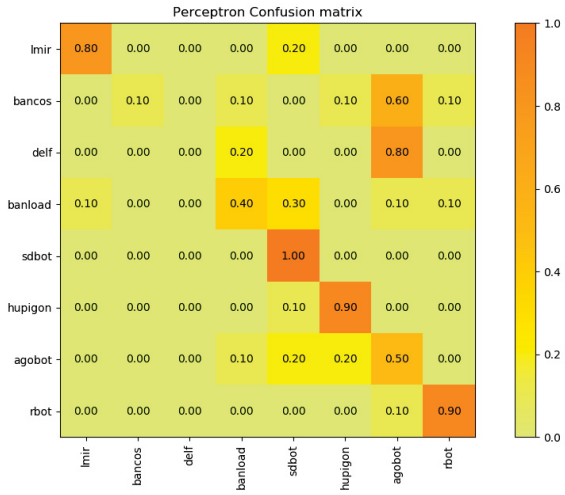


Figura A.189: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

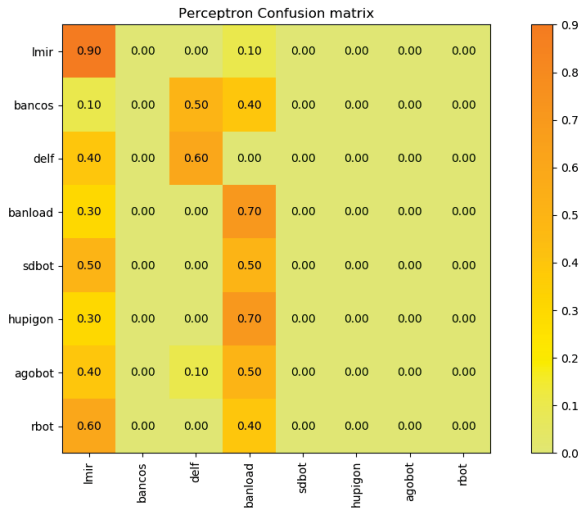


Figura A.190: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

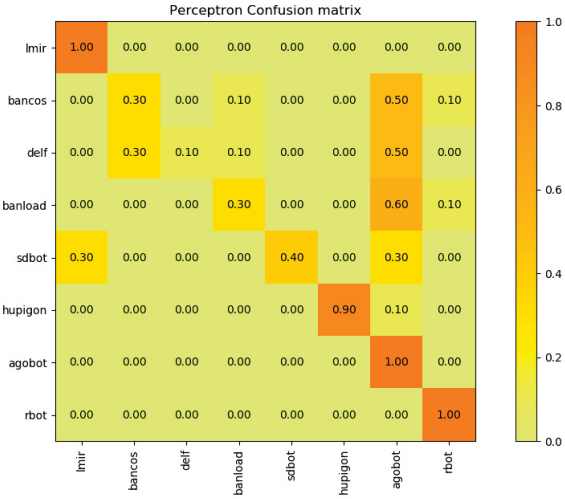


Figura A.191: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

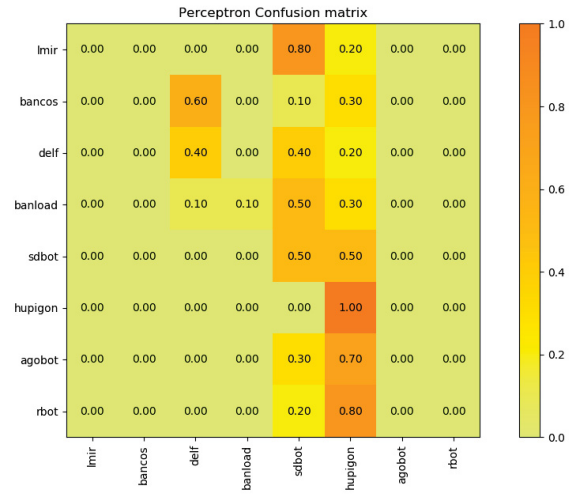


Figura A.192: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **MLP.**

Figura A.193: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

Figura A.194: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 no subconjunto do Malimg Dataset.

Figura A.195: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

Figura A.196: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 no subconjunto do Malimg Dataset.

- **CNN.**

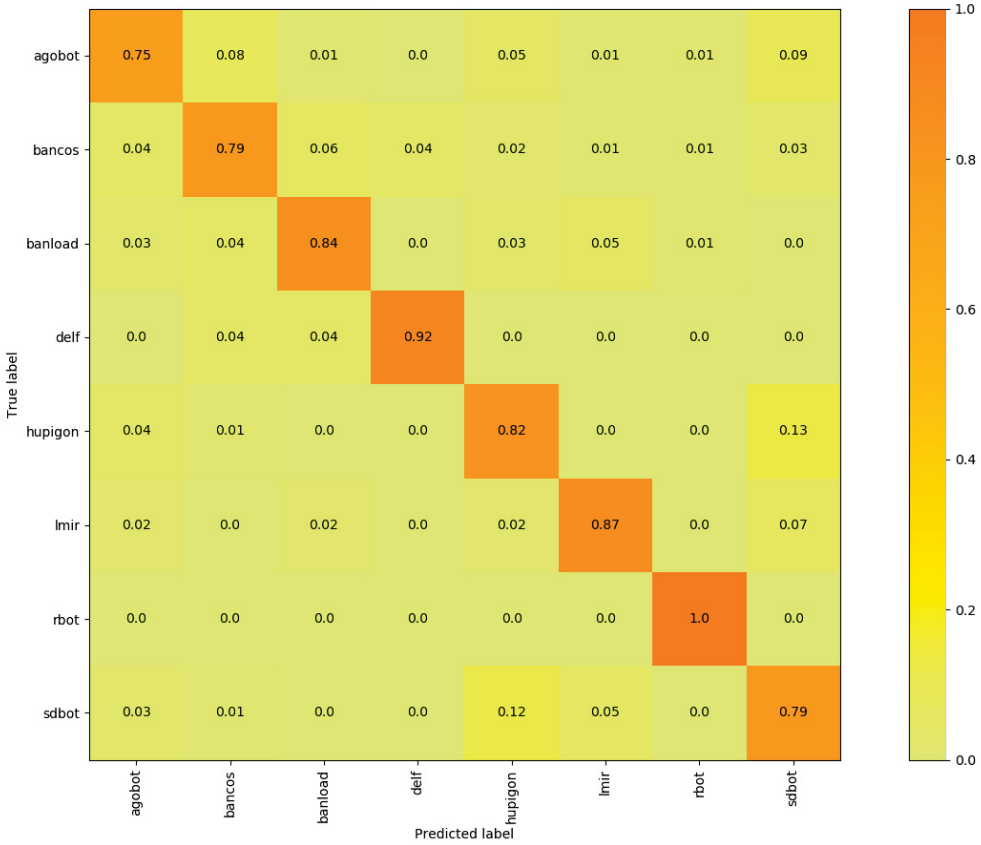


Figura A.197: Matriz de confusão do classificador CNN no subconjunto do Malimg Dataset.

Para os experimentos realizados neste subconjunto do *dataset* local, ainda nota-se que algumas famílias normalmente são classificadas incorretamente, inclusive as mesmas apontadas no subconjunto com 10 famílias, mas também se observa que as famílias `Hupigon` e `Lmir` normalmente atingem mais de 80% de acurácia. Apesar de algumas famílias serem similares entre si, há algumas famílias que possuem características bastante singulares, que os diferentes classificadores e conseguem distinguir. Porém sabe-se que a quantidade de famílias de *malware* existentes é muito superior as utilizadas nos experimentos e podem existir outras famílias mais similares a essas que atingem alta taxa de classificação. Essa técnica não pode ser considerada viável por apresentar uma boa taxa de classificação apenas em um cenário com amostras selecionadas e apenas para algumas famílias. Quando é aumentada a complexidade ela falha, como pode ser observado em um cenário com binários ofuscados, na próxima seção.

A.3 BINÁRIOS OFUSCADOS

Para demonstrar a habilidade de classificação das técnicas usando análise de textura em binários ofuscados, foram realizados experimentos em diferentes compressores. Os resultados deles usando `ZIP` podem ser observados nas matrizes de confusão abaixo.

- **KNN.**

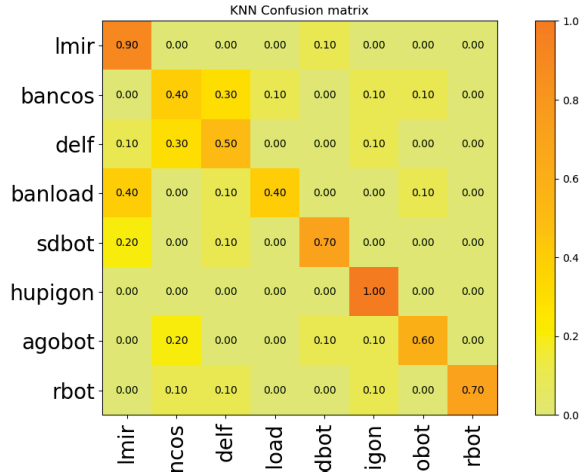


Figura A.198: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com ZIP.

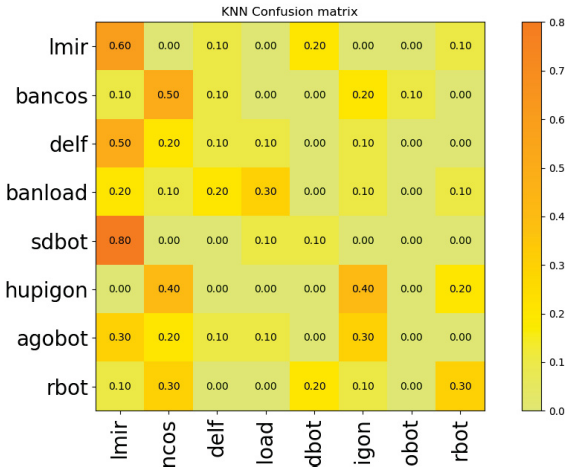


Figura A.199: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com ZIP.

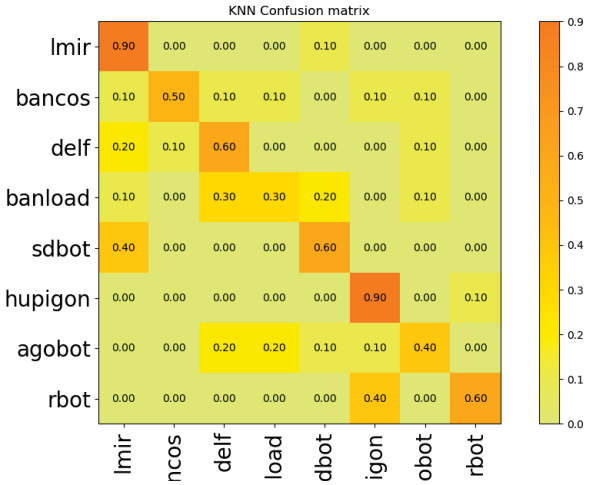


Figura A.200: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com ZIP.

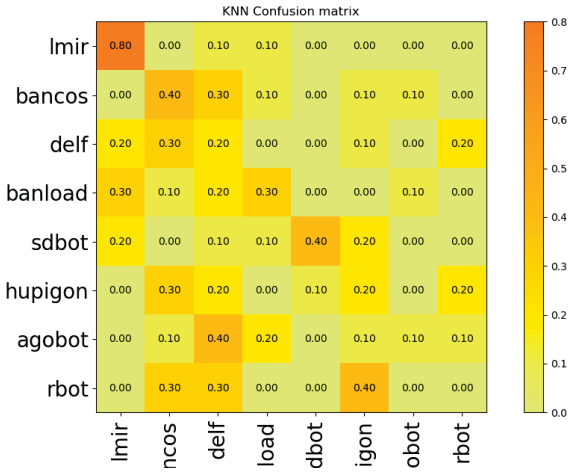


Figura A.201: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com ZIP.

- **Decision Trees.**

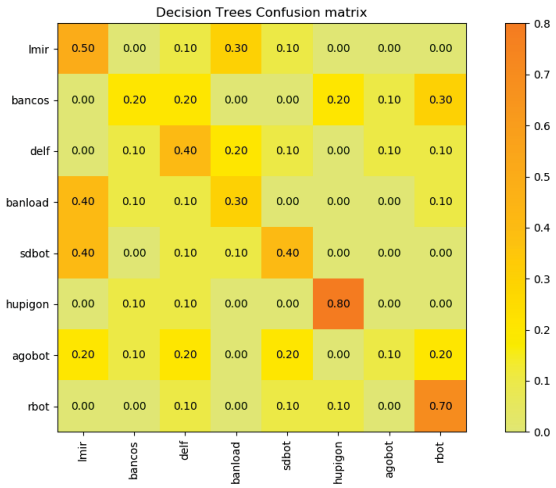


Figura A.202: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `ZIP`.

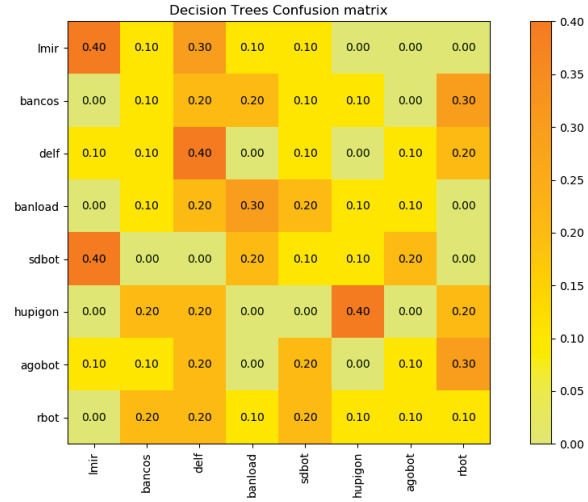


Figura A.203: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `ZIP`.

Figura A.204: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

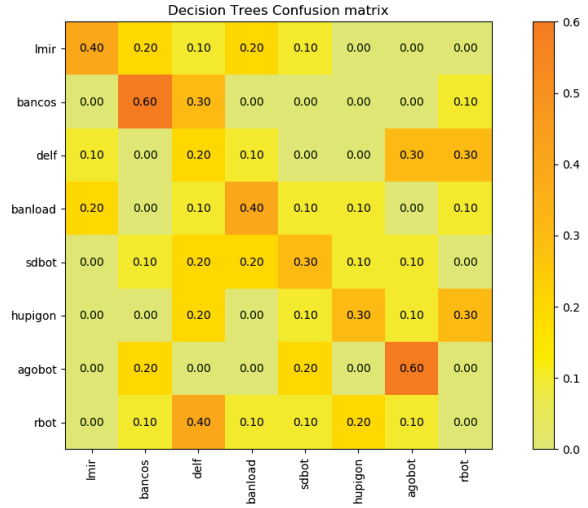


Figura A.205: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

- **Random Forest.**

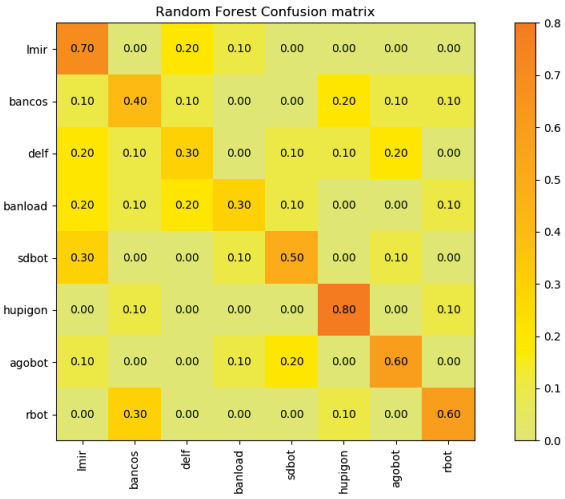


Figura A.206: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com ZIP.

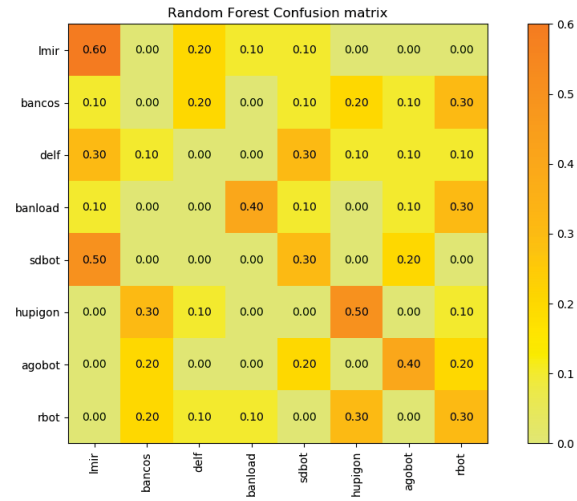


Figura A.207: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com ZIP.

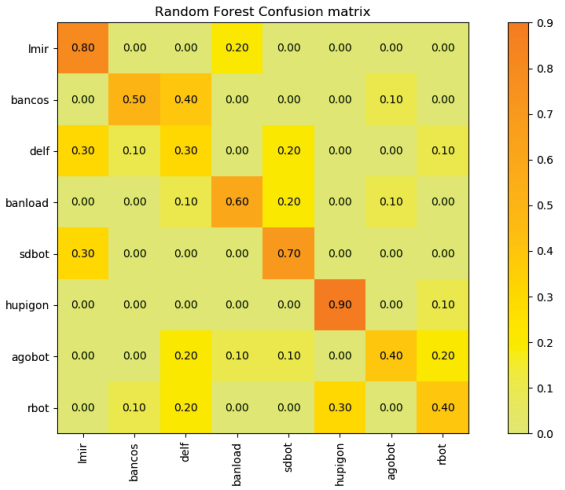


Figura A.208: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com ZIP.

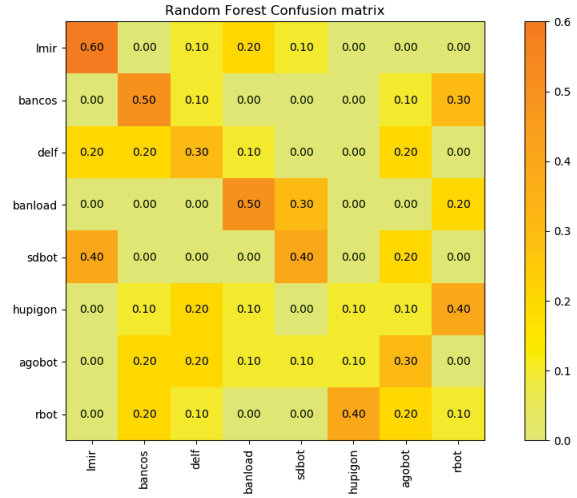


Figura A.209: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com ZIP.

- **Nearest Centroid**

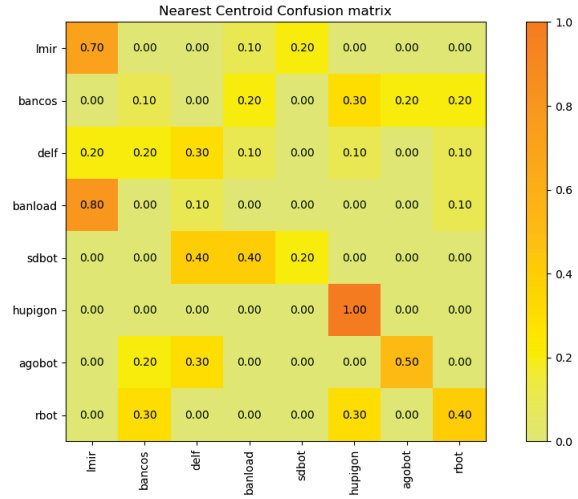


Figura A.210: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com ZIP.

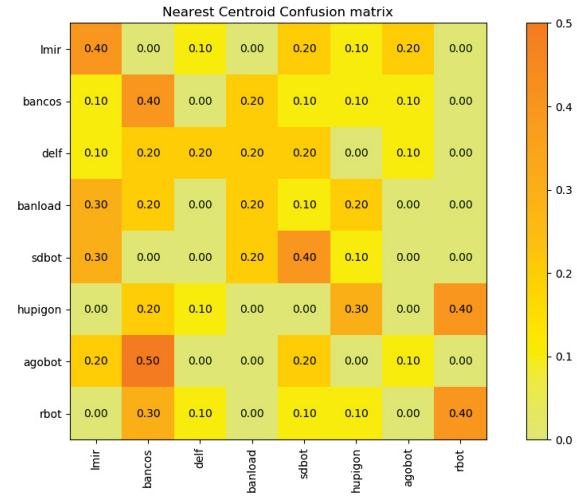


Figura A.211: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com ZIP.

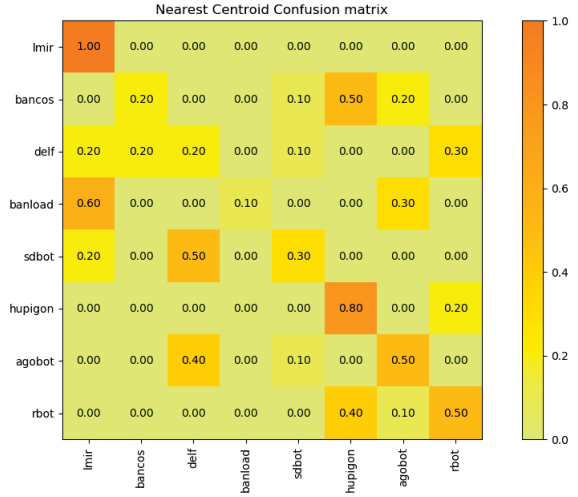


Figura A.212: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com ZIP.

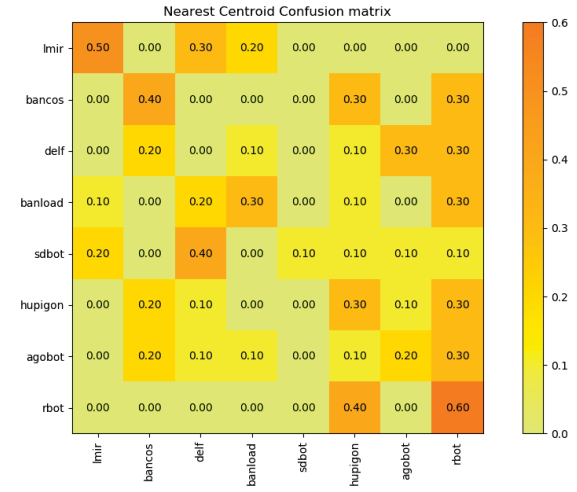


Figura A.213: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com ZIP.

- **SVM.**

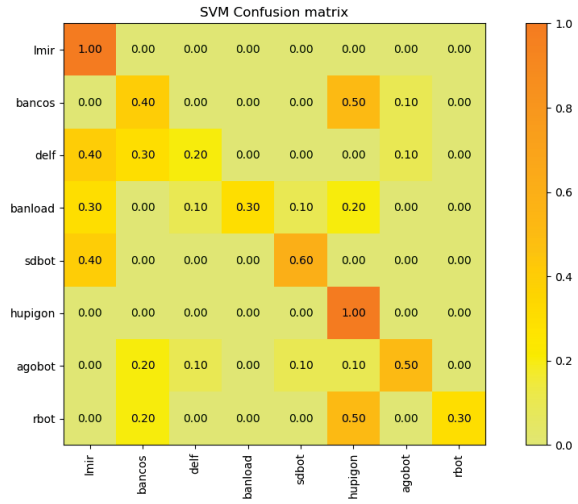


Figura A.214: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com `ZIP`.

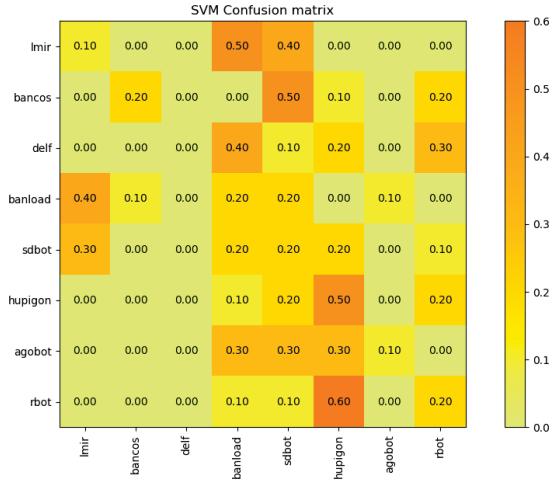


Figura A.215: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com `ZIP`.

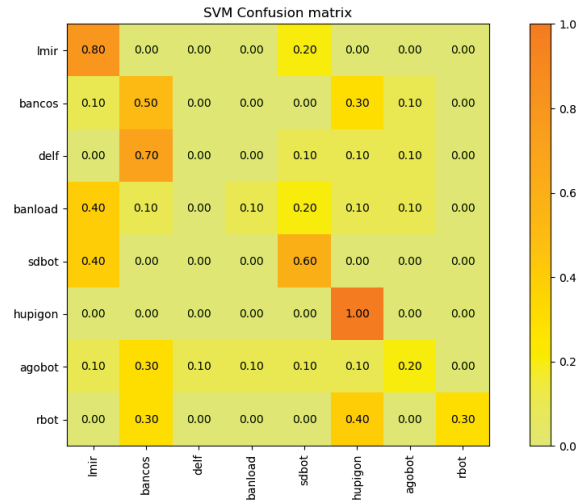


Figura A.216: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

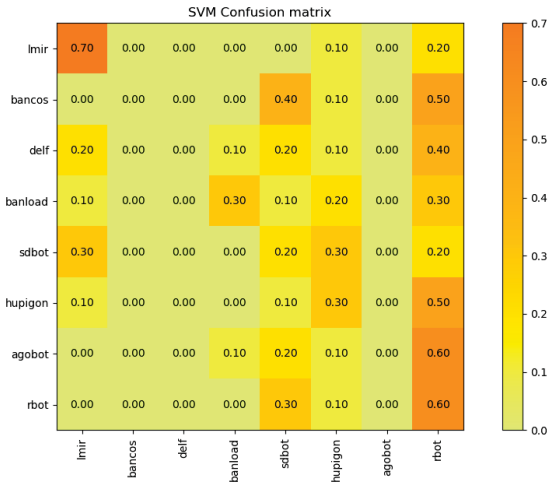


Figura A.217: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

- **SGD.**

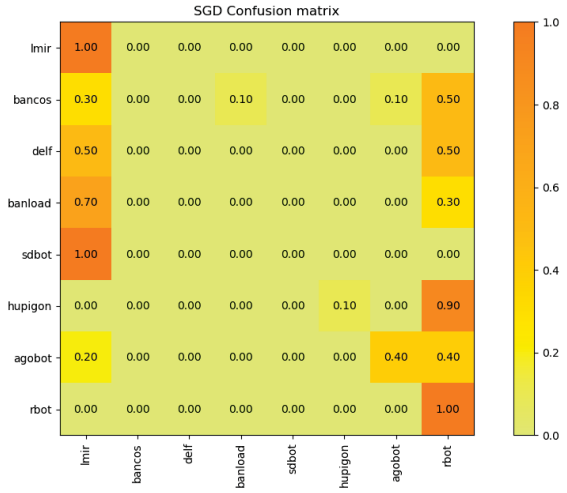


Figura A.218: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `ZIP`.

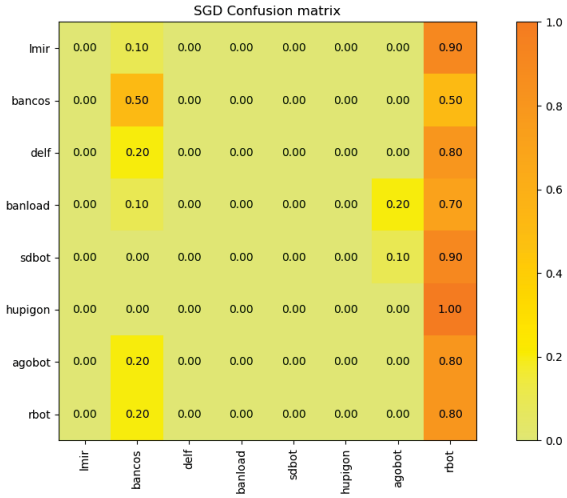


Figura A.219: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `ZIP`.

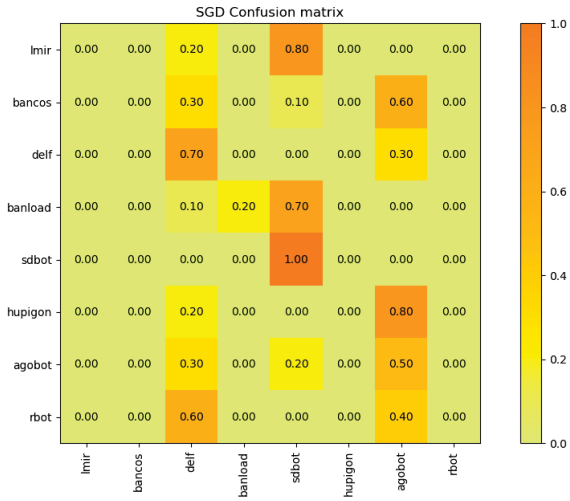


Figura A.220: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

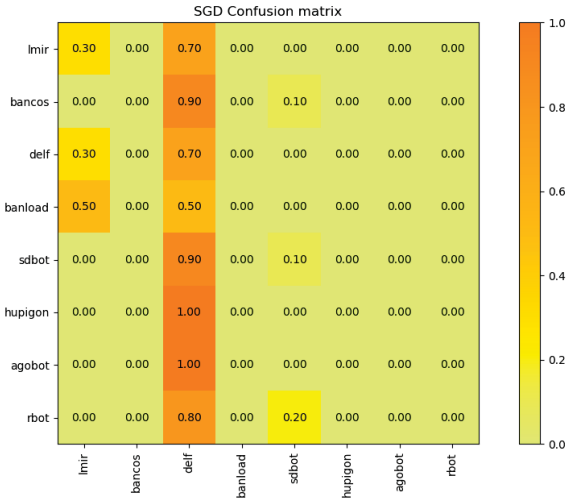


Figura A.221: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

- **Perceptron.**

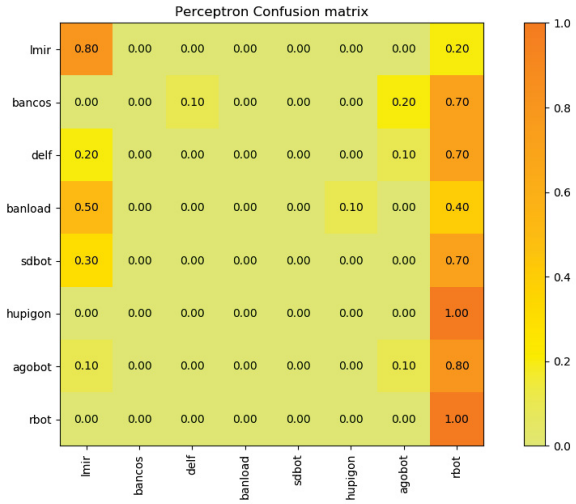


Figura A.222: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `ZIP`.

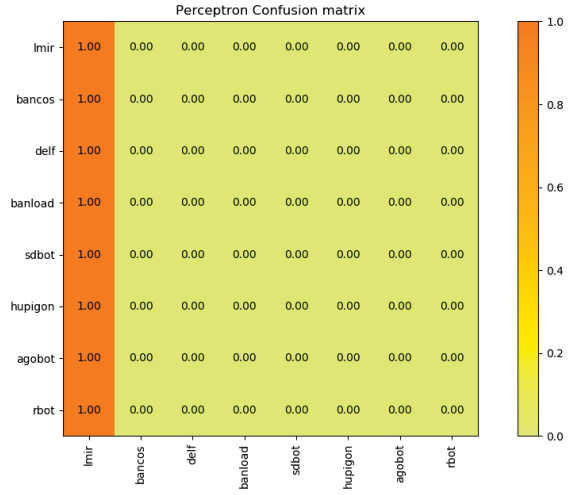


Figura A.223: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `ZIP`.

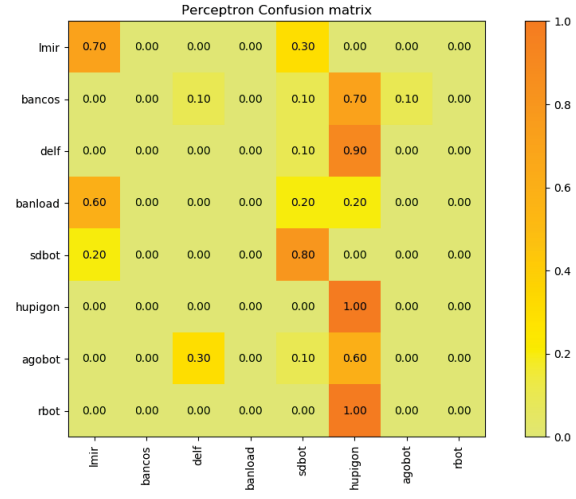


Figura A.224: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

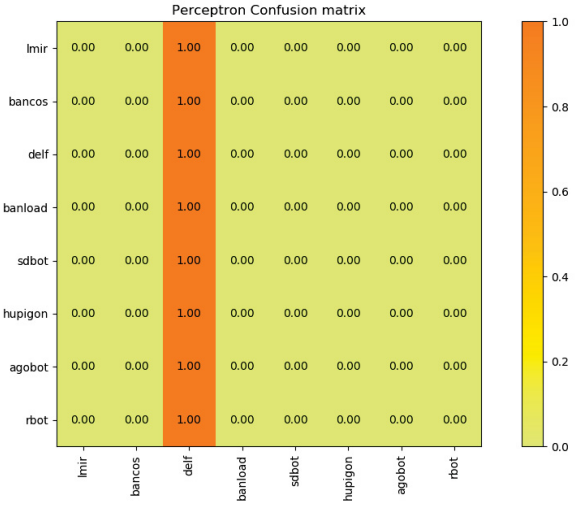


Figura A.225: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

- **MLP.**

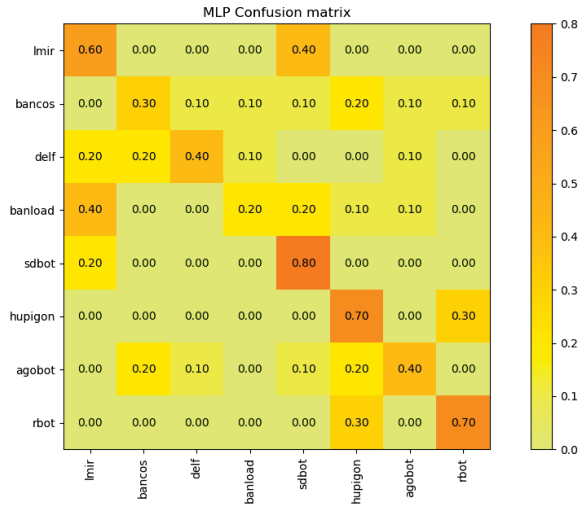


Figura A.226: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `ZIP`.

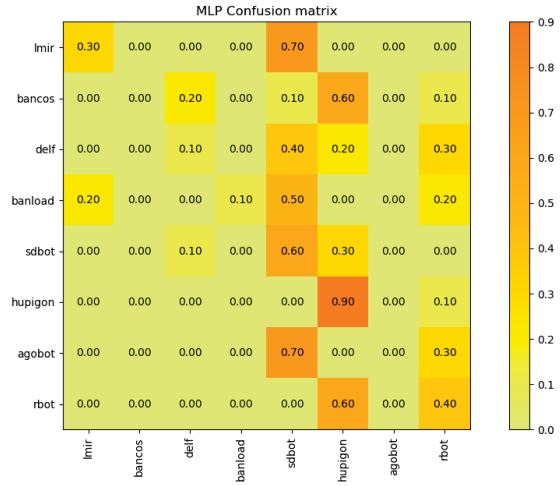


Figura A.227: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `ZIP`.

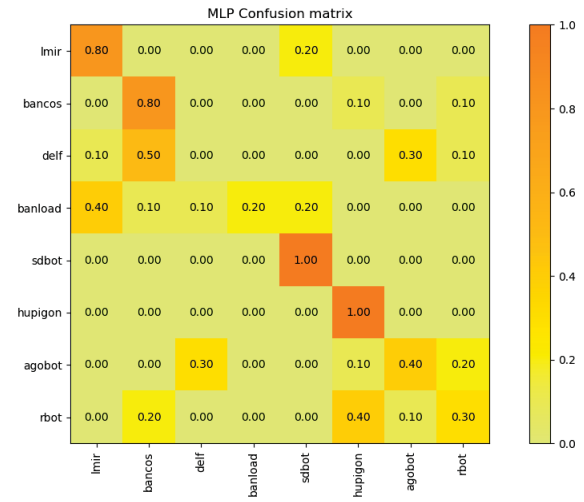


Figura A.228: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

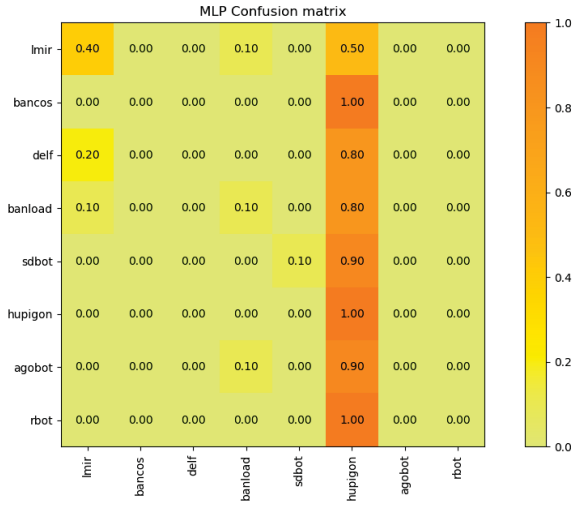


Figura A.229: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `ZIP`.

- **CNN.**

Figura A.230: Matriz de confusão do classificador CNN em um subconjunto do *dataset* local ofuscado com `ZIP`.

No mesmo *dataset* usado nos últimos experimentos, usando `ZIP`, as famílias `Hupigon` e `Lmir` continuam apresentando os melhores resultados, porém a família `Delf`, que apresentava um resultado relativamente bom antes da ofuscação, apresenta um dos piores nesses experimentos. Mas fica claro que a classificação dos binários ofuscados ficou com taxas bem inferiores.

A seguir foram os algoritmos foram aplicados no mesmo *dataset* porém ofuscado com `TAR.GZ` e as matrizes de confusão geradas são apresentadas abaixo.

- **KNN.**

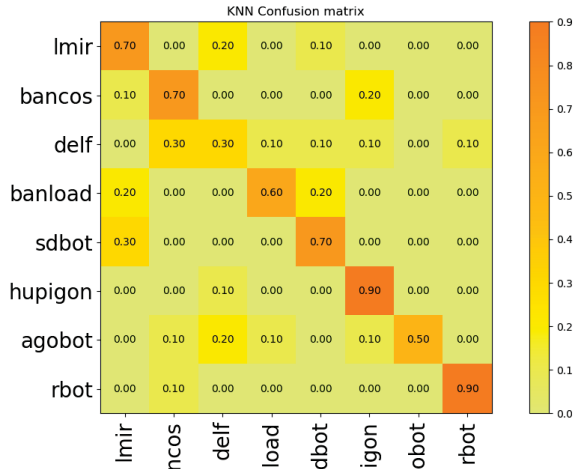


Figura A.231: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

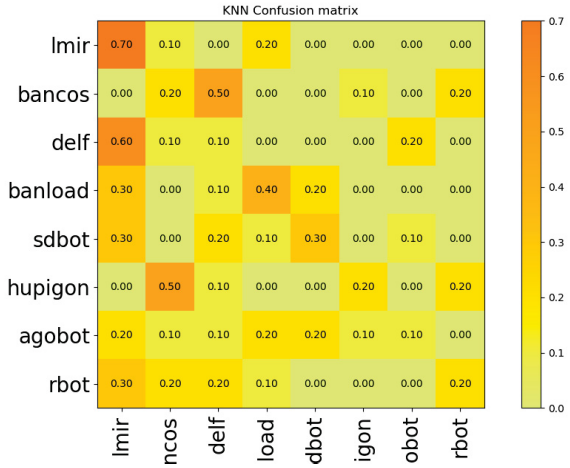


Figura A.232: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

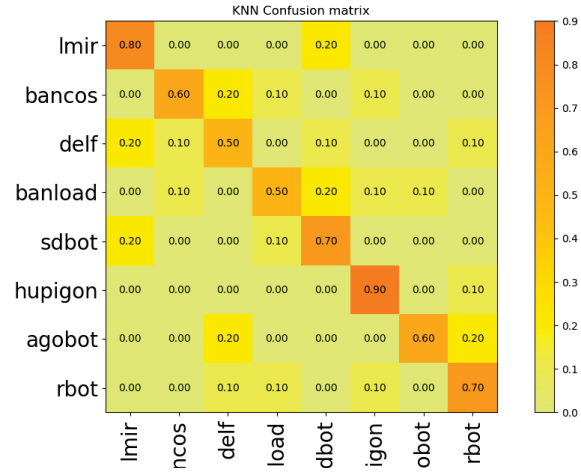


Figura A.233: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

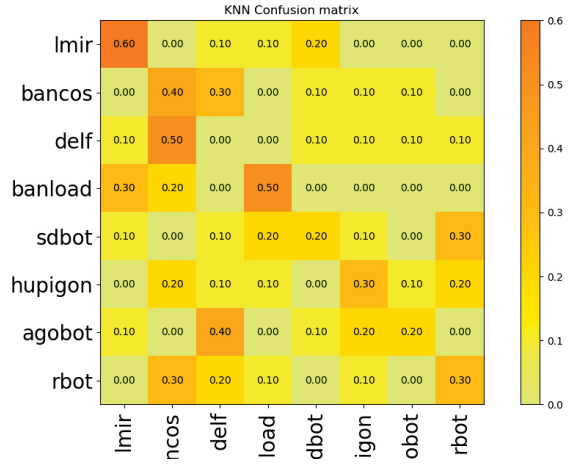


Figura A.234: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

- **Decision Trees.**

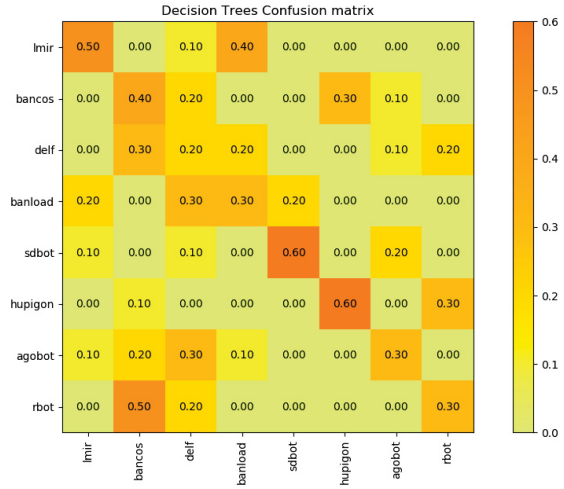


Figura A.235: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

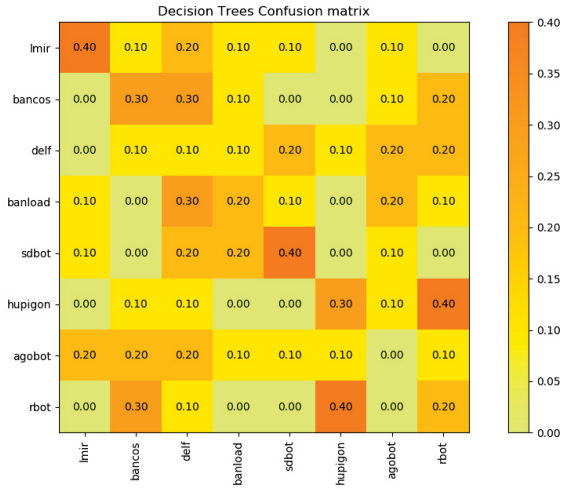


Figura A.236: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

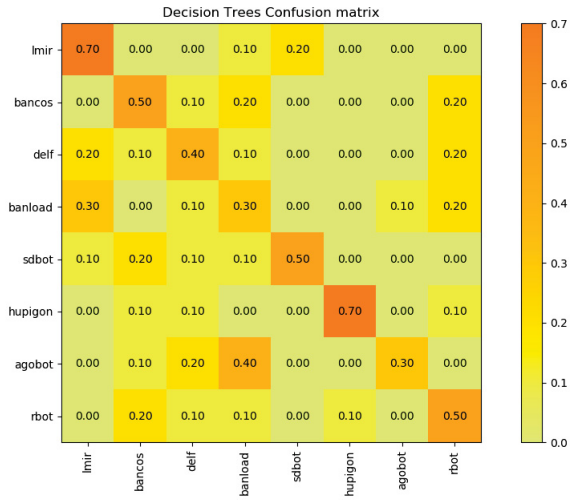


Figura A.237: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

Figura A.238: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

- **Random Forest.**

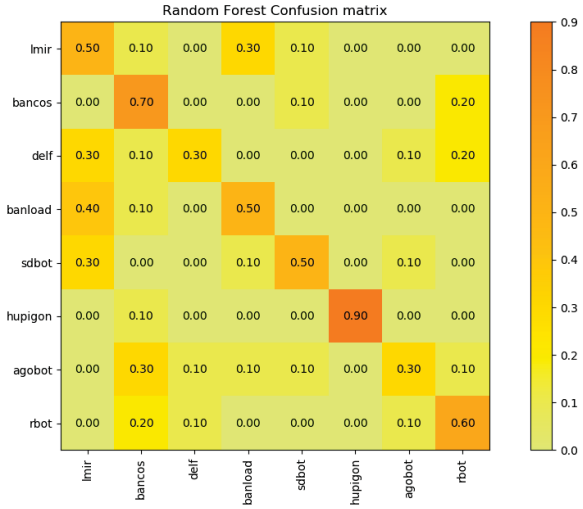


Figura A.239: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

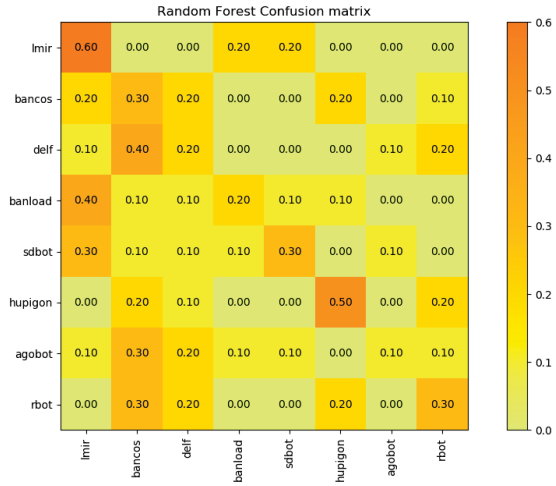


Figura A.240: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

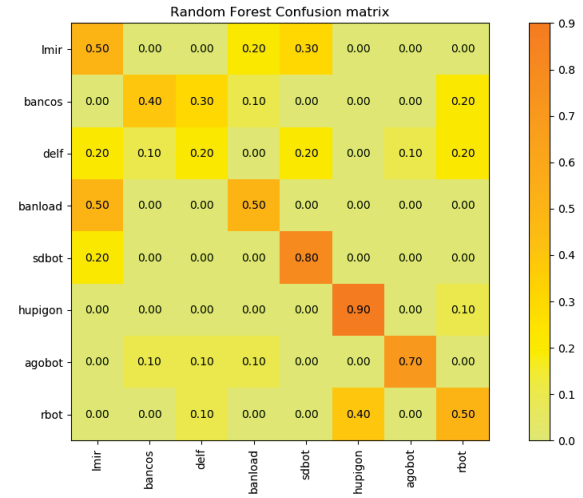


Figura A.241: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

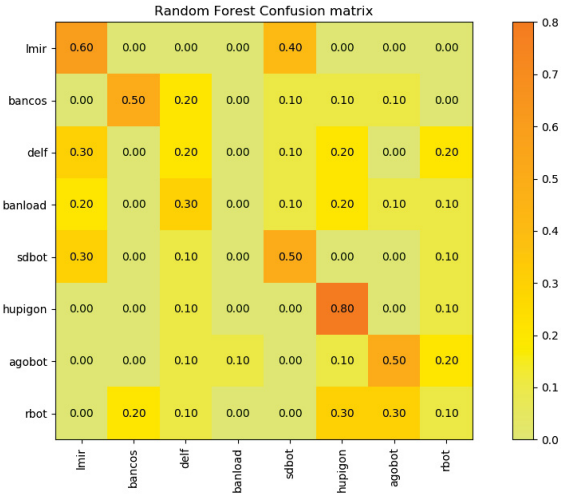


Figura A.242: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

- **Nearest Centroid.**

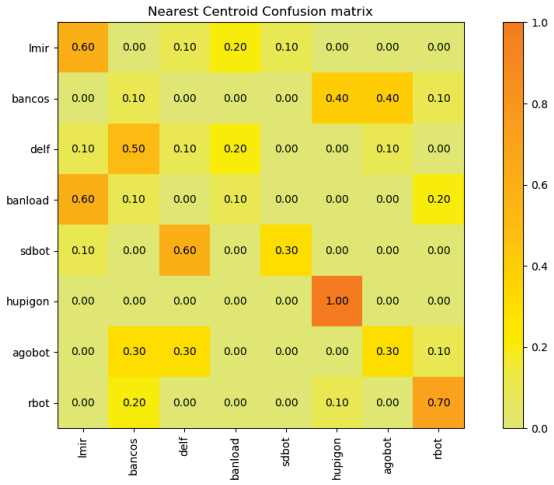


Figura A.243: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

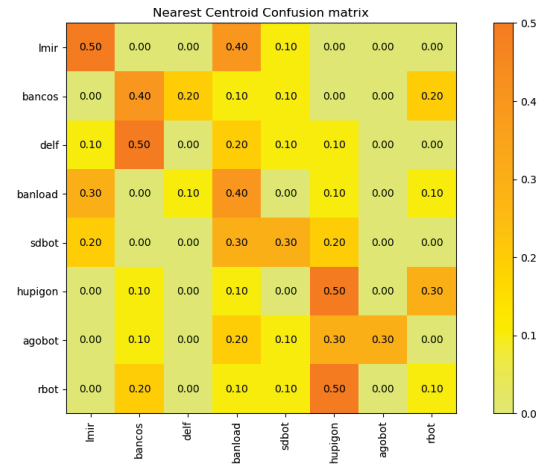


Figura A.244: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

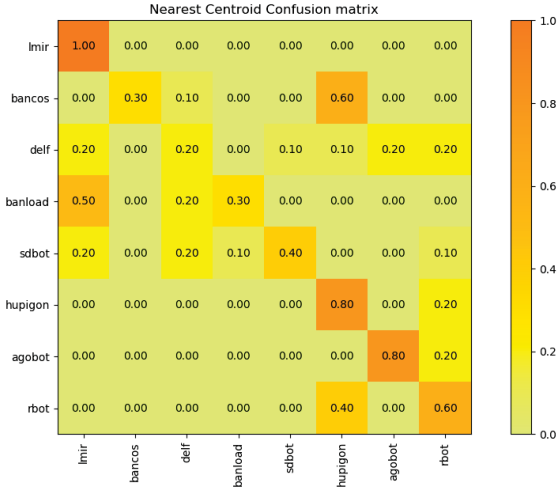


Figura A.245: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

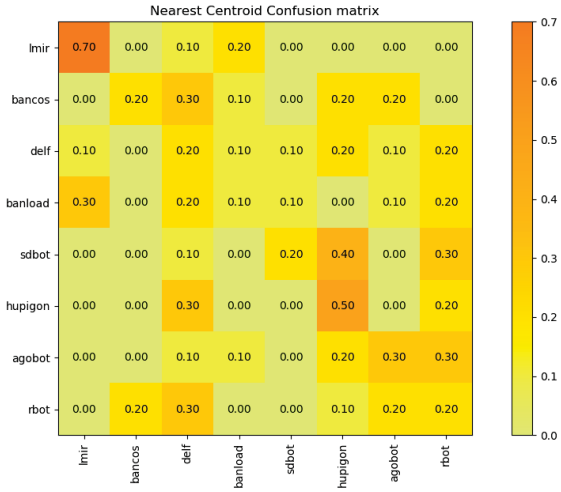


Figura A.246: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

- **SVM.**

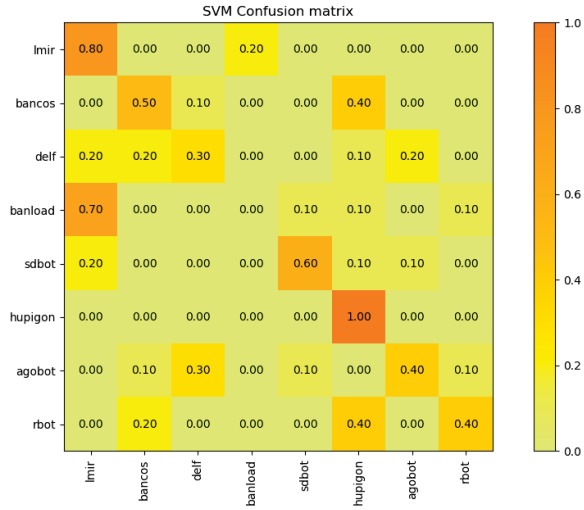


Figura A.247: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

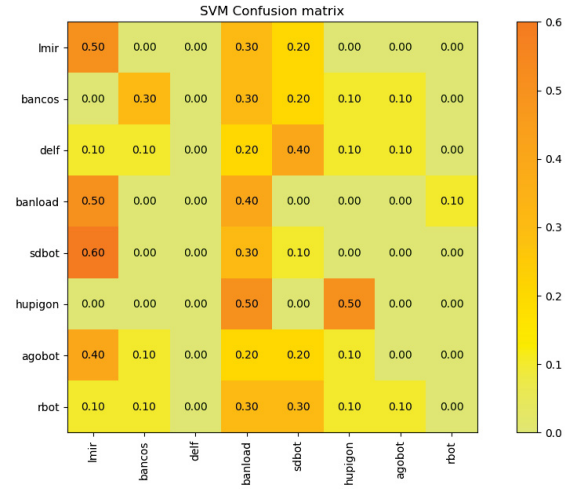


Figura A.248: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

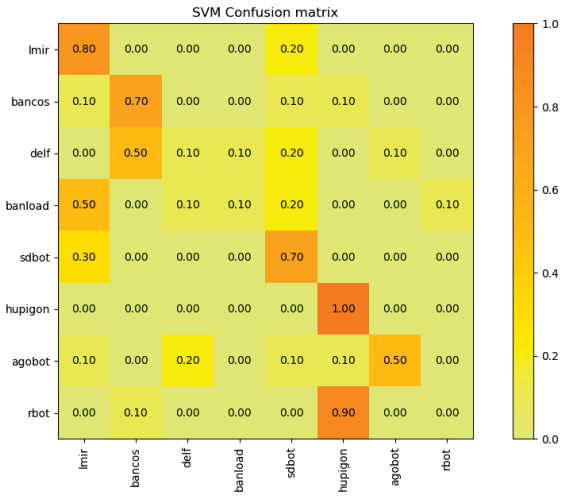


Figura A.249: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

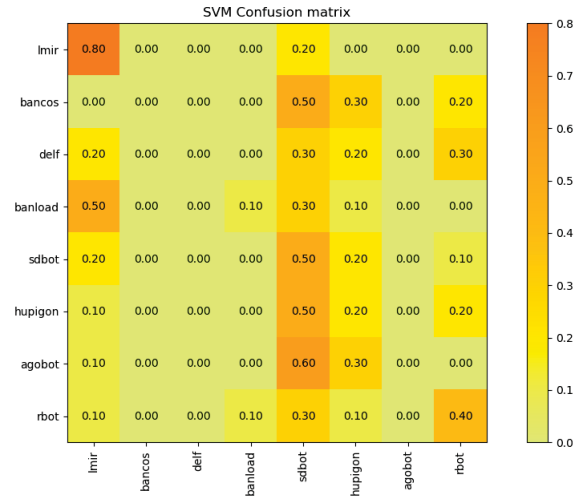


Figura A.250: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

- **SGD.**

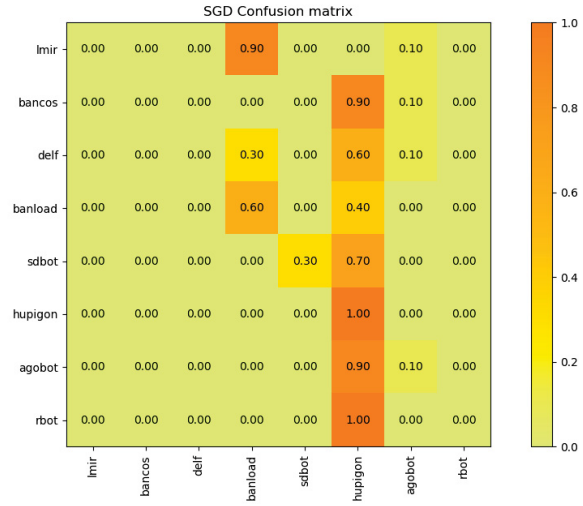


Figura A.251: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

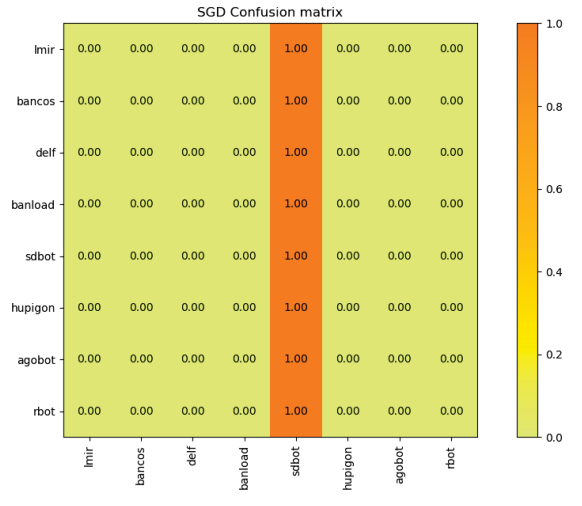


Figura A.252: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

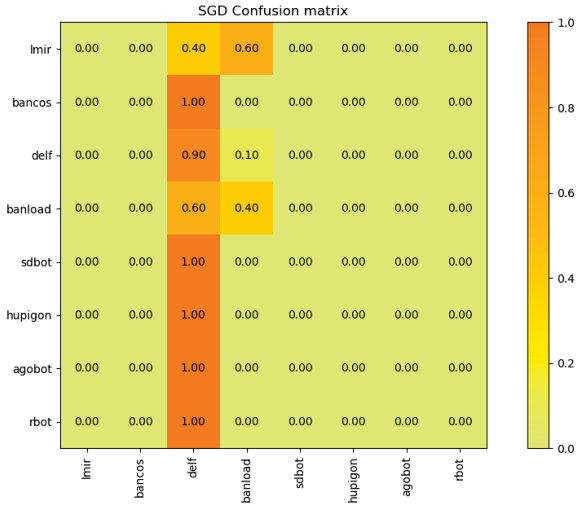


Figura A.253: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

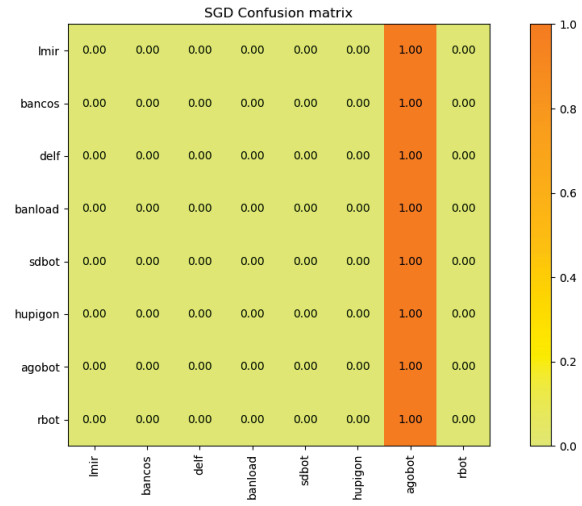


Figura A.254: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

- **Perceptron.**

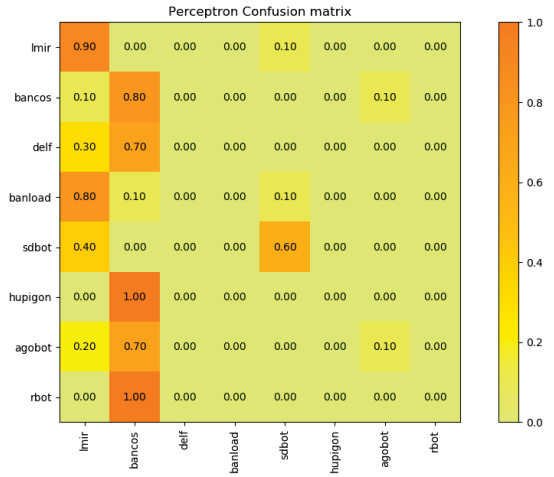


Figura A.255: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

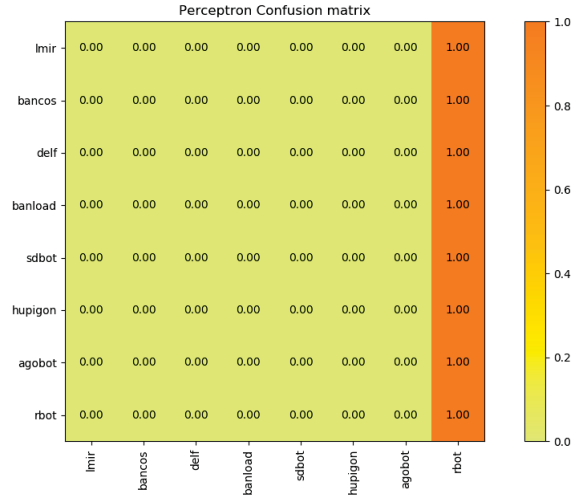


Figura A.256: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

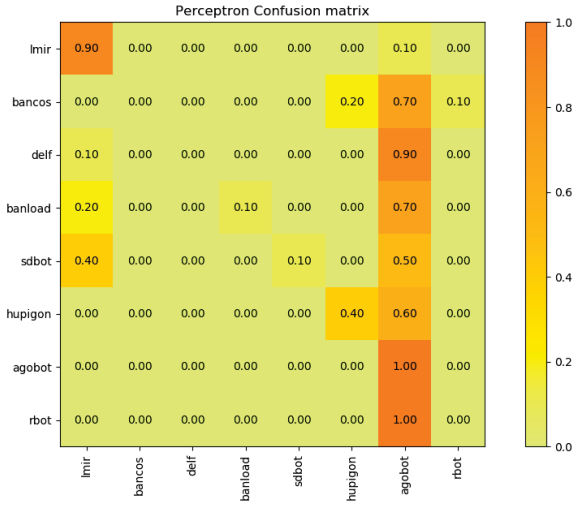


Figura A.257: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

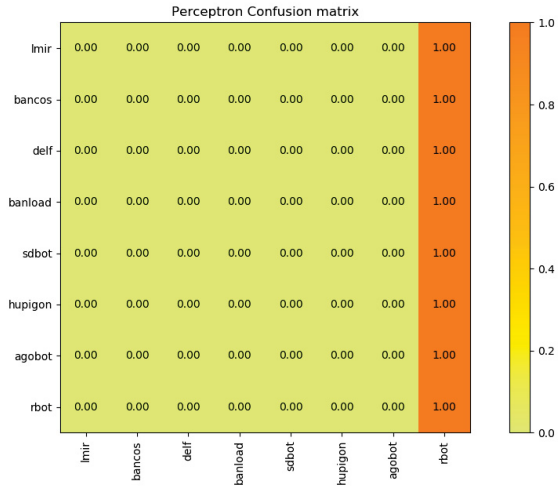


Figura A.258: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

- **MLP.**

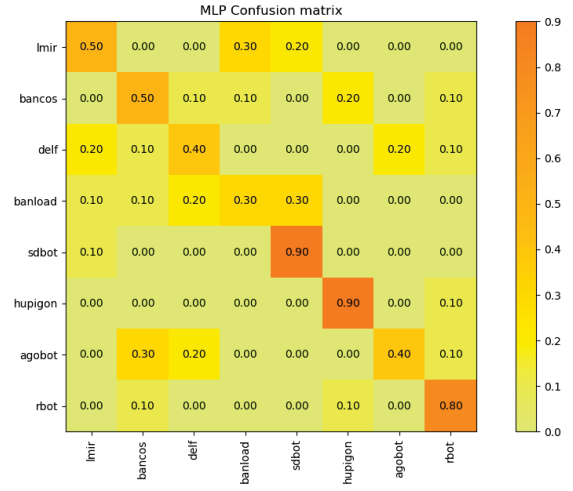


Figura A.259: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

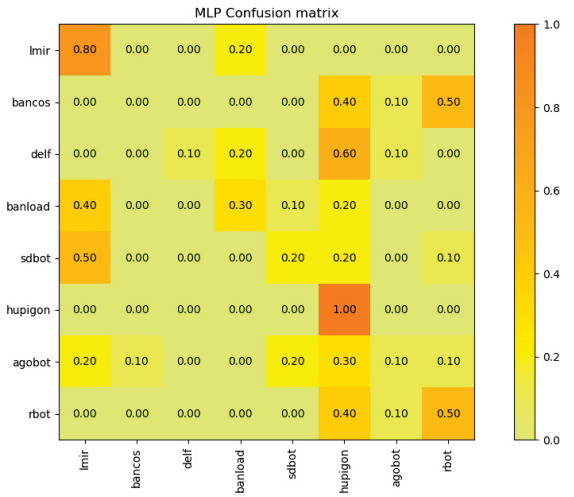


Figura A.260: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

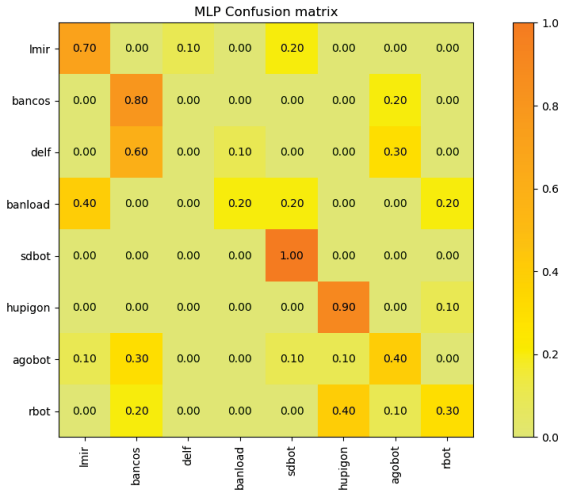


Figura A.261: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

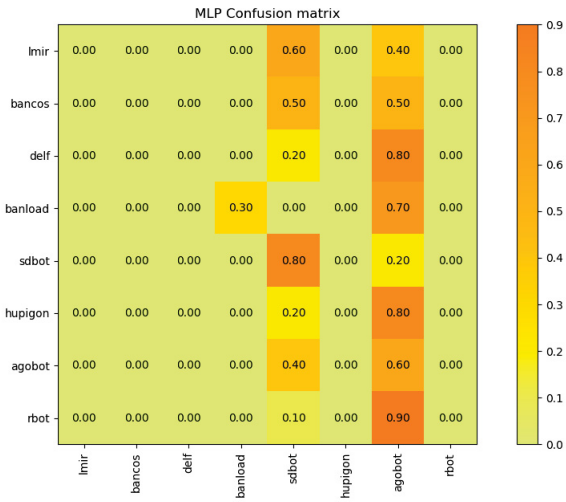


Figura A.262: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

- **CNN.**

Figura A.263: Matriz de confusão do classificador CNN em um subconjunto do *dataset* local ofuscado com `TAR.GZ`.

Nesses experimentos a família `Delf` apresentou o pior resultado na maioria das vezes e as mesmas famílias do experimento anterior apresentaram os melhores resultados. Porém todas elas foram classificadas erradas em alguns casos e as taxas de classificação reduziram significativamente.

Após esses experimentos foi utilizado o `UPX` para ofuscação e as matrizes de confusão são exibidas abaixo.

- **KNN.**

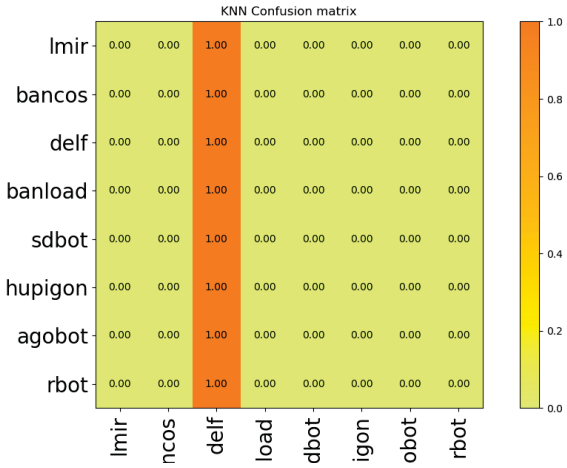


Figura A.264: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com UPX.

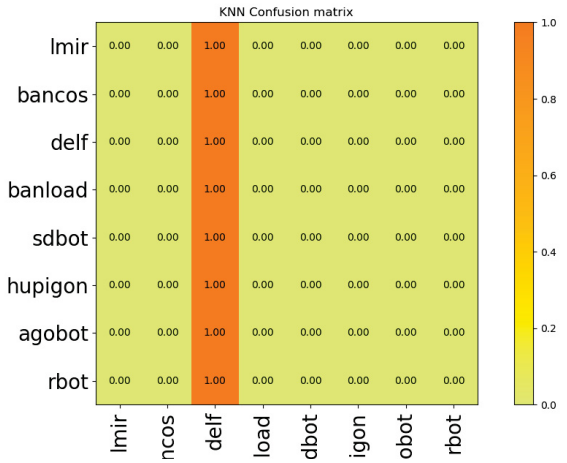


Figura A.265: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com UPX.

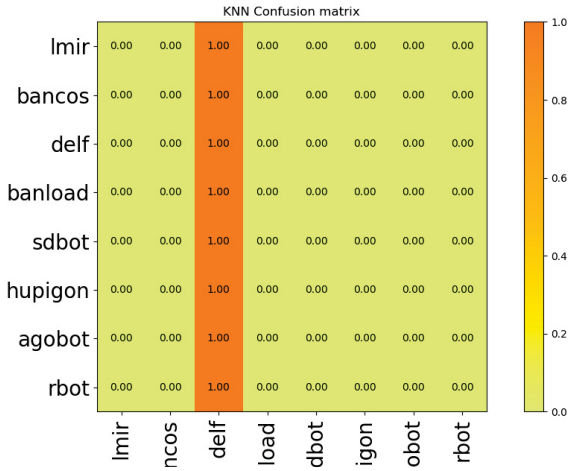


Figura A.266: Matriz de confusão do classificador KNN utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com UPX.

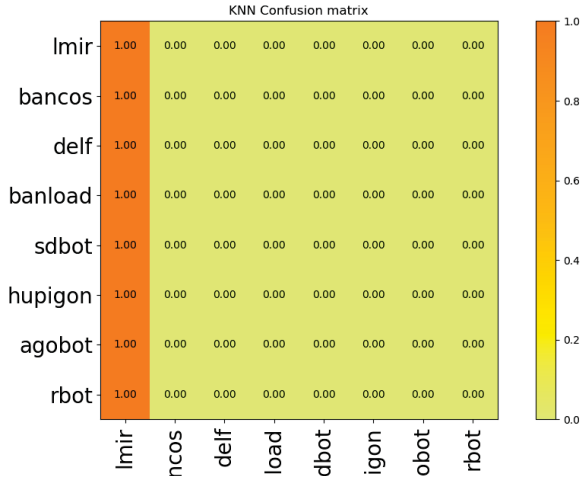


Figura A.267: Matriz de confusão do classificador KNN utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com UPX.

- **Decision Trees.**

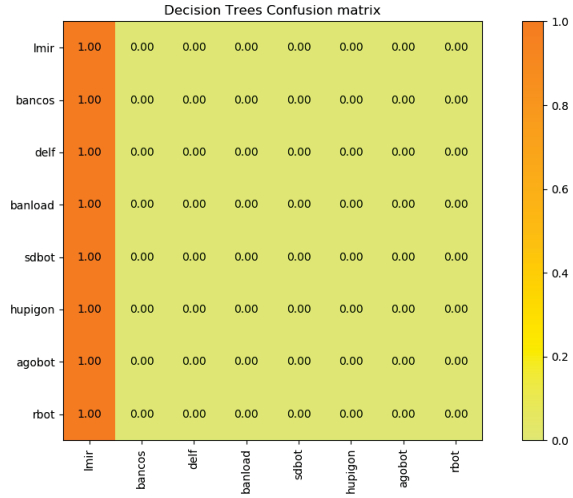


Figura A.268: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

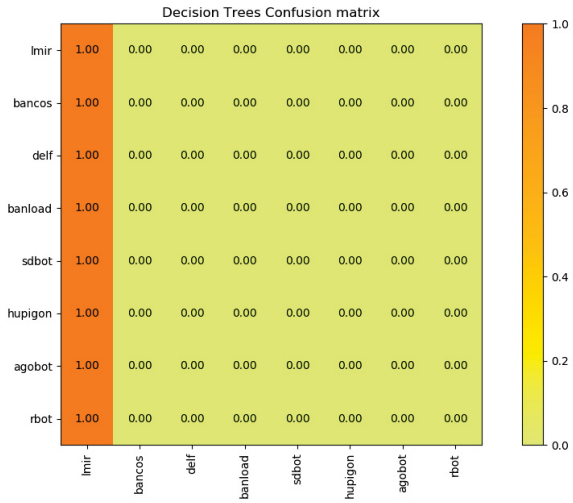


Figura A.269: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

Figura A.270: Matriz de confusão do classificador *Decision Trees* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

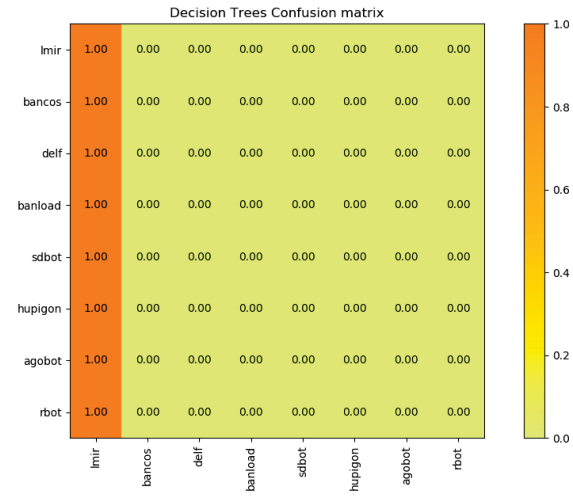


Figura A.271: Matriz de confusão do classificador *Decision Trees* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

- **Random Forest.**

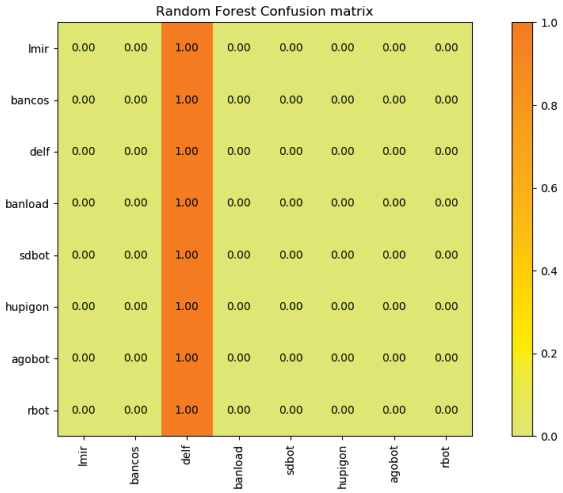


Figura A.272: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

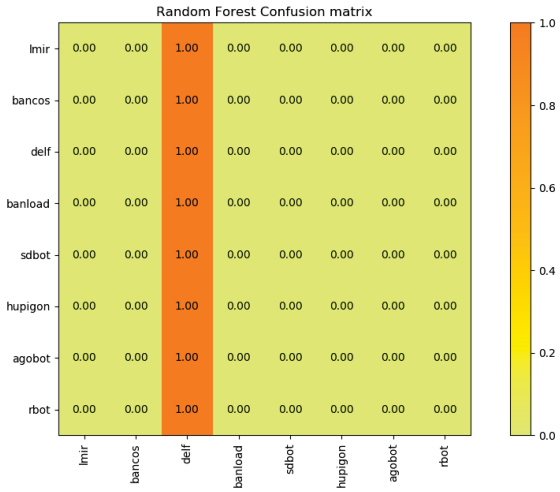


Figura A.273: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

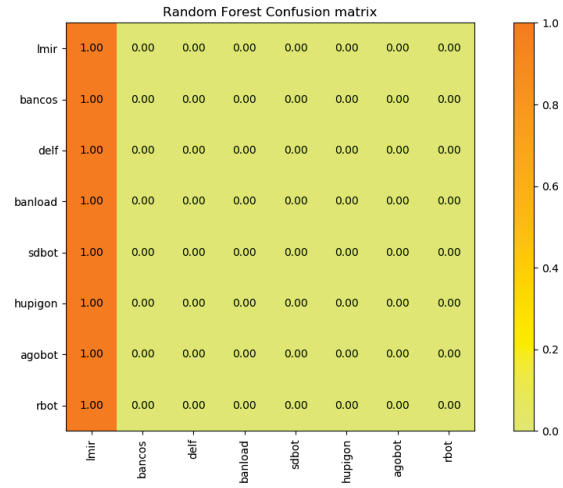


Figura A.274: Matriz de confusão do classificador *Random Forest* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

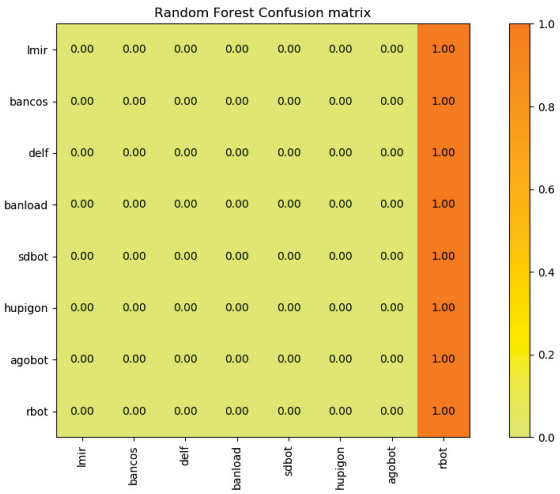


Figura A.275: Matriz de confusão do classificador *Random Forest* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

- **Nearest Centroid**

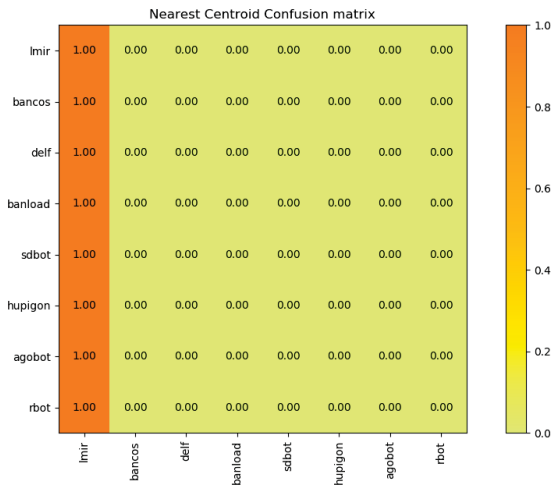


Figura A.276: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

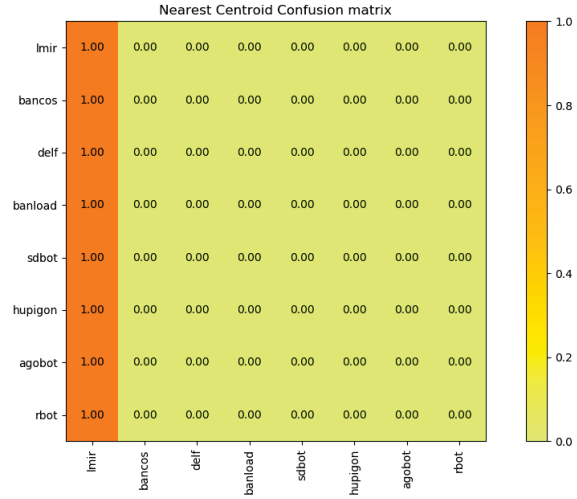


Figura A.277: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

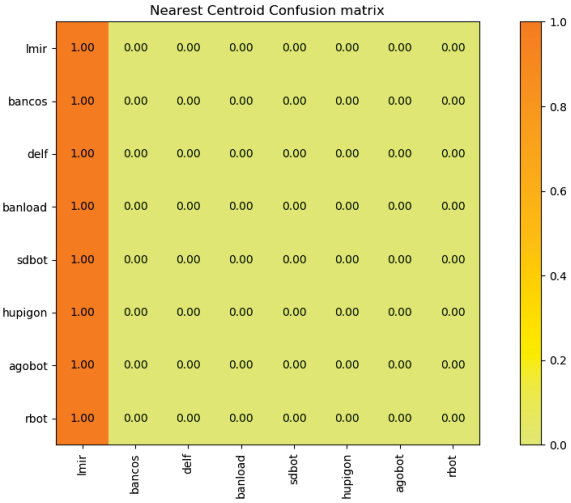


Figura A.278: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

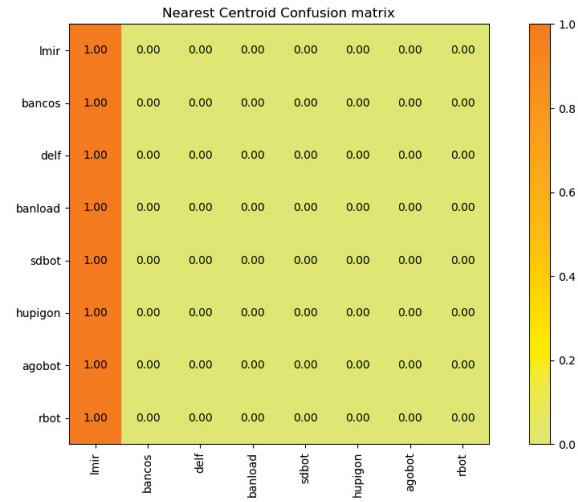


Figura A.279: Matriz de confusão do classificador *Nearest Centroid* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

- **SVM.**

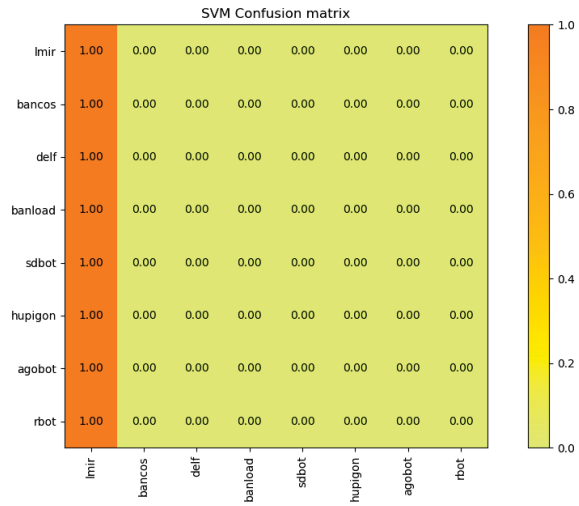


Figura A.280: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com UPX.

Figura A.281: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 64x64 em um subconjunto do *dataset* local ofuscado com UPX.

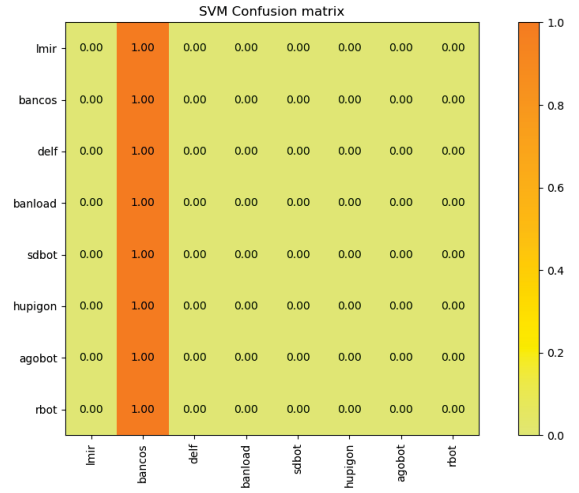


Figura A.282: Matriz de confusão do classificador SVM utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com UPX.

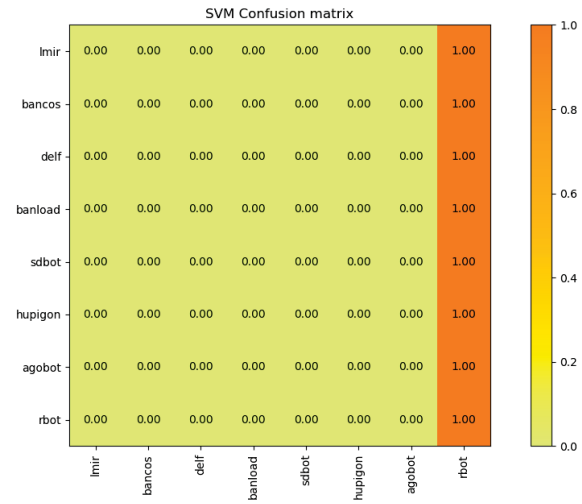


Figura A.283: Matriz de confusão do classificador SVM utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com UPX.

- **SGD.**

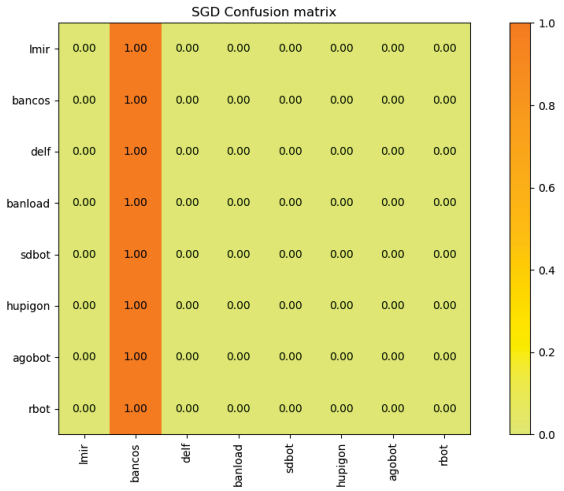


Figura A.284: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com UPX.

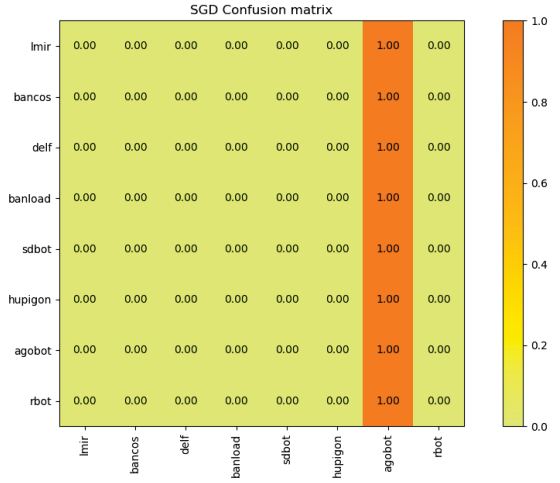


Figura A.285: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com UPX.

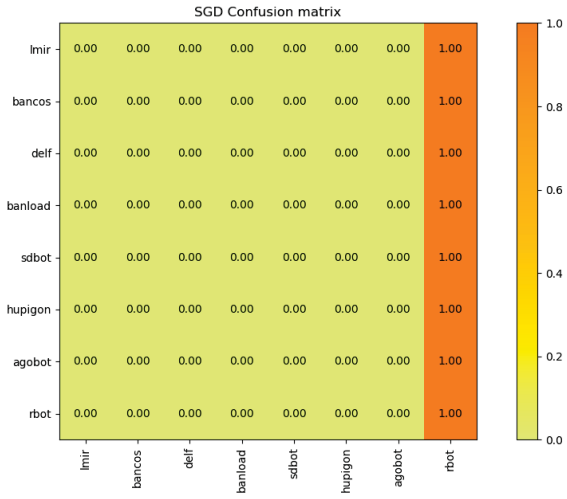


Figura A.286: Matriz de confusão do classificador SGD utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com UPX.

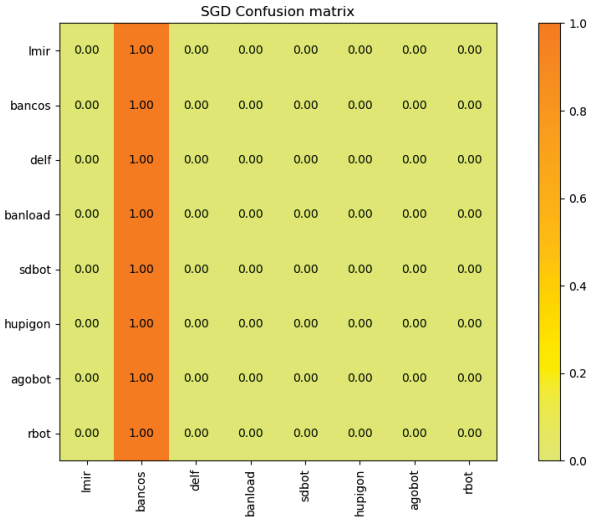


Figura A.287: Matriz de confusão do classificador SGD utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com UPX.

- **Perceptron.**

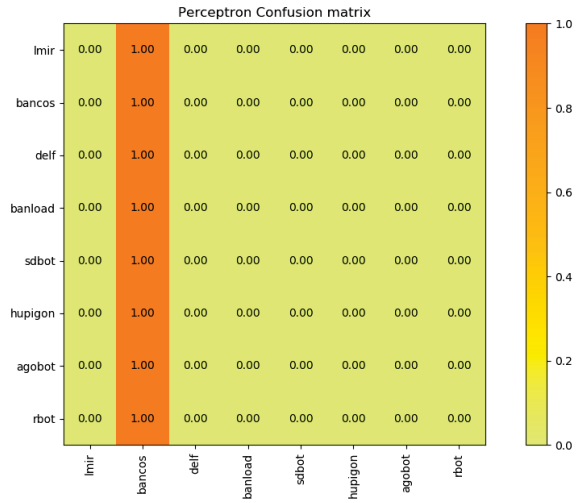


Figura A.288: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

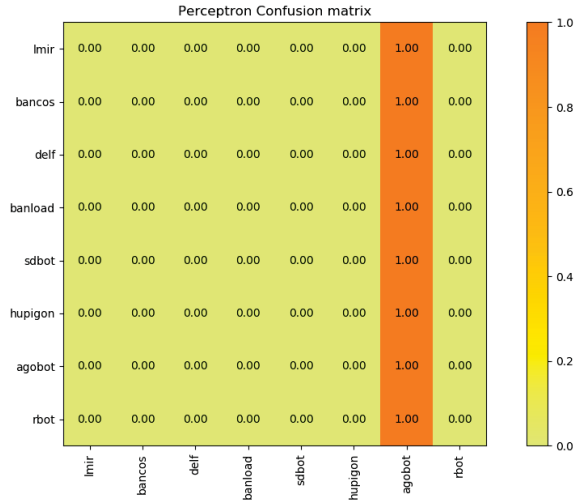


Figura A.289: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

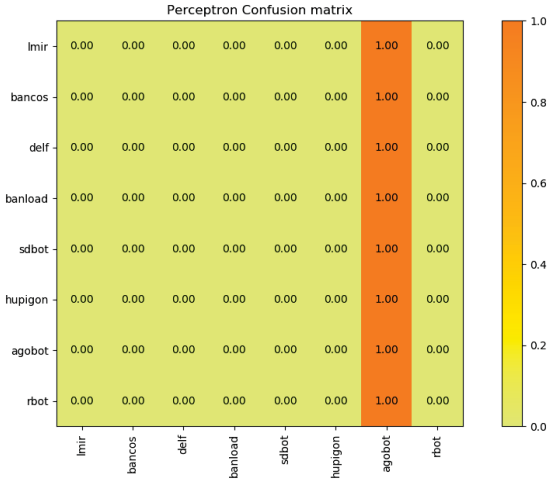


Figura A.290: Matriz de confusão do classificador *Perceptron* utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

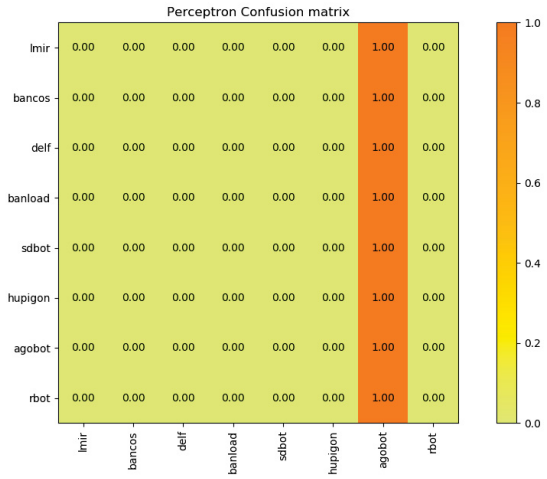


Figura A.291: Matriz de confusão do classificador *Perceptron* utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

- **MLP.**

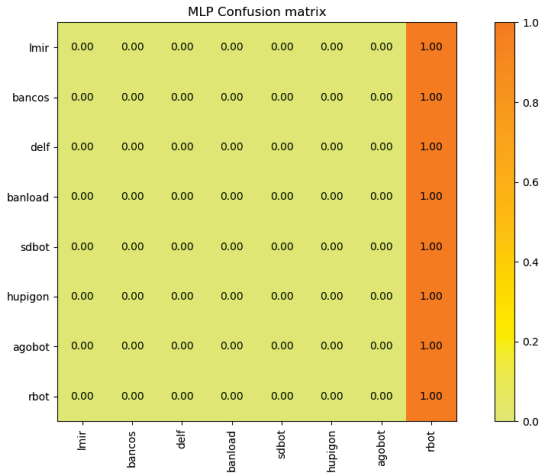


Figura A.292: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

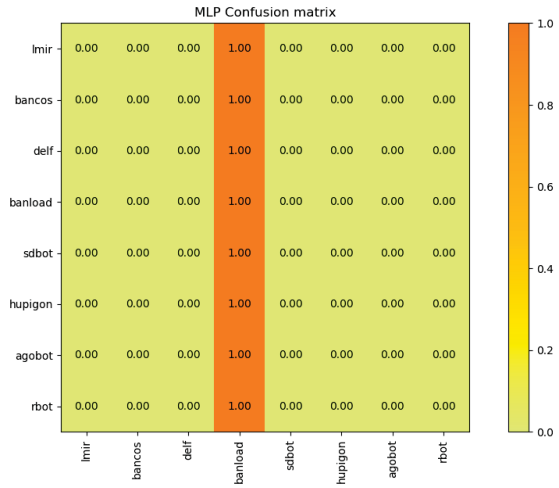


Figura A.293: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 32x32 em um subconjunto do *dataset* local ofuscado com `UPX`.

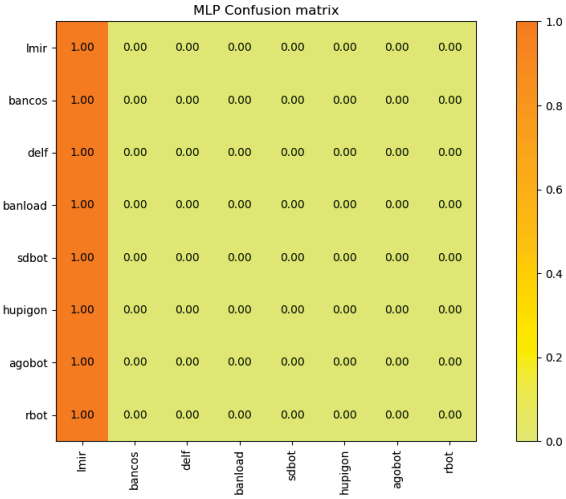


Figura A.294: Matriz de confusão do classificador MLP utilizando descritor GIST e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

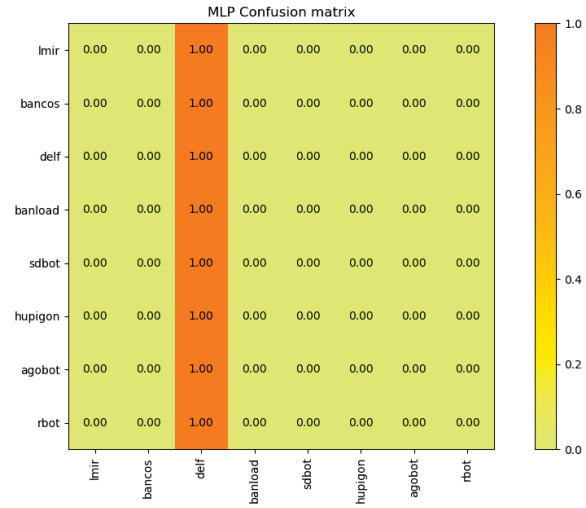


Figura A.295: Matriz de confusão do classificador MLP utilizando descritor LBP e redimensionamento de 128x128 em um subconjunto do *dataset* local ofuscado com `UPX`.

- **CNN.**

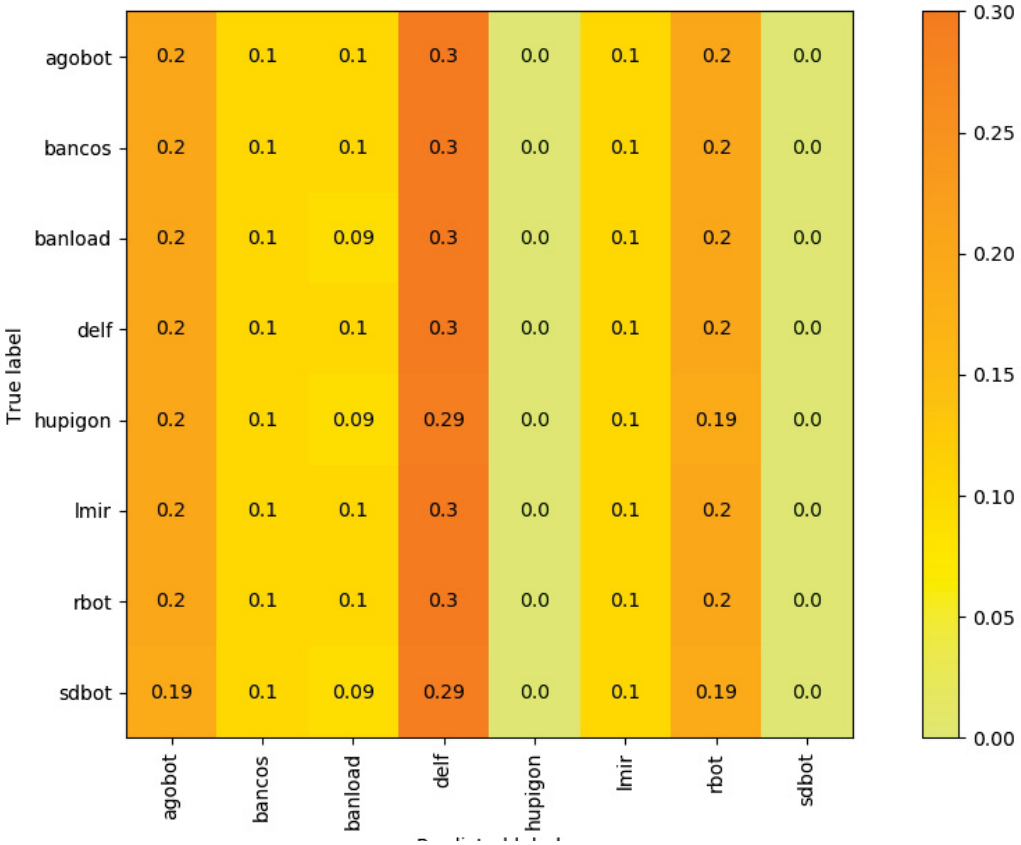


Figura A.296: Matriz de confusão do classificador CNN em um subconjunto do *dataset* local ofuscado com `UPX`.

Usando `UPX` os classificadores agruparam todas as amostras na mesma família na maioria dos casos. Usando análise de textura não é possível diferenciar esses binários, mostrando que essas técnicas de classificação não são resilientes a essas técnicas de ofuscação.